

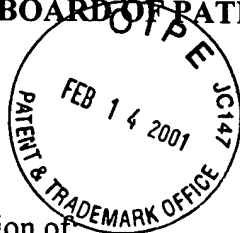
IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

#16
KWS
2-28-01
10/13
RECEIVED

Group Art Unit: 2767
Examiner: J. Darrow

FEB 21 2001

Technology Center 2100



FEE ENCLOSED: \$155-
Please charge any further
fee to our Deposit Account
No. 18-0002

In Re Patent Application of:

Applicant(s) : SCHEIDT et al.

Serial No. : 09/023,672

Confirmation No. 7278

Filed : February 13, 1998

For : CRYPTOGRAPHIC KEY SPLIT
COMBINER

Attorney Ref. : STS 119

BRIEF ON APPEAL

Commissioner for Patents
Washington, D.C. 20231

INTRODUCTION

This is an Appeal from a final Office Action dated July 27, 2000. This Brief on Appeal is being filed in triplicate as required by 37 CFR §1.192(a), together with the fee of \$155.00 as required by 37 CFR §1.17(c). If no remittance is attached or if any additional fees are required, please charge any deficiency to our deposit account, No. 18-0002, and notify us accordingly.

Claims 1-69 are pending in the application. In the final Action, the Examiner rejected claims 1, 2, 4, 32-36, 38, and 67-69 under 35 USC §102(b) as being anticipated

by *Hirsch*, U.S. Patent No. 5,276,738. The Examiner also rejected claims 3 and 37 under 35 USC §103(b) as being unpatentable over *Hirsch*, in view of *Albert et al.*, United States Patent No. 5,627,894. The Examiner also rejected claims 5 and 39 under 35 USC §103(b) as being unpatentable over *Hirsch*, in view of *Thomlinson et al.*, United States Patent No. 5,778,069. The Examiner also rejected claims 6, 7, 9-11, 13-16, 40, 41, 43-45, and 47-50 under 35 USC §103(b) as being unpatentable over *Hirsch*, in view of *Ming et al.*, United States Patent No. 5,710,815. The Examiner also rejected claims 8 and 42 under 35 USC §103(b) as being unpatentable over *Hirsch*, in view of *Ming et al.*, and further in view of *Anshel et al.*, U.S. Patent No. 5,751,808. The Examiner also rejected claims 12 and 46 under 35 USC §103(b) as being unpatentable over *Hirsch*, in view of *Ming et al.*, and further in view of *Albert et al.*, U.S. Patent No. 5,627,738. The Examiner also rejected claims 17 and 51 under 35 USC §103(b) as being unpatentable over *Hirsch*, in view of *Ming et al.*, and further in view of *Anshel et al.*, U.S. Patent No. 5,751,808. The Examiner also rejected claims 18, 20-24, 52, and 54-58 under 35 USC §103(b) as being unpatentable over *Hirsch*, in view of *Anshel et al.* The Examiner also rejected claims 19 and 53 under 35 USC §103(b) as being unpatentable over *Hirsch*, in view of *Anshel et al.*, and further in view of *Albert et al.* The Examiner also rejected claims 25, 27-31, 59, and 61-65 under 35 USC §103(b) as being unpatentable over *Hirsch*, in view of *Tomko et al.*, United States Patent No. 5,541,994. The Examiner also rejected claims 26 and 60 under 35 USC §103(b) as being unpatentable over *Hirsch*, in view of *Tomko et al.*, and further in view of *Albert et al.*

In response to these rejections, Appellant filed a Notice of Appeal on August 10, 2000. A copy of the claims subject to this Appeal is attached as Appendix A.

REAL PARTY IN INTEREST

The present application has been assigned to TecSec, Incorporated, of Vienna, Virginia, which is the real party in interest.

RELATED APPEALS AND INTERFERENCES

To the best of the undersigned attorney's knowledge and belief, no other appeals or interferences are or have been filed that will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

STATUS OF THE CLAIMS

This application was filed on February 13, 1998, with claims 1-69.

Claims 1-69, as originally filed, are appealed.

STATUS OF AMENDMENTS

A Notice of Appeal was filed on August 10, 2000 in response to the Examiner's Final Action of July 27, 2000. There have been no amendments in this application.

SUMMARY OF THE INVENTION

The present invention is a system for formulating cryptographic keys that are used to encrypt plaintext messages and to decrypt ciphertext messages. That is, an original plaintext message is encrypted according to an encrypt text/key relation using an encrypt key, to create a ciphertext message. Likewise, an authorized entity having a proper

decrypt key can apply the decrypt key to the ciphertext message according to a decrypt text/key relation to create a new plaintext message that corresponds to the original plaintext message. The claimed invention is a system for formulating these keys. The keys are composed of two or more components, called key splits, each of which may be provided by a different source. These different sources of key splits are key split generators, which generate the key splits based on seed data. These individual splits are randomized to form the complete cryptographic key. The complete key may take the form of a stream of symbols, a group of symbol blocks, a key matrix, or any other form suitable for use by the particular encryption scheme of the encryption system.

ISSUES ON APPEAL

1. Whether claims 1, 2, 4, 32-36, 38, and 67-69 are anticipated under 35 USC §102(b) by *Hirsch*, U.S. Patent No. 5,276,738.
2. Whether claims 3 and 37 are unpatentable within the meaning of 35 USC §103(b), as being unpatentable over *Hirsch*, in view of *Albert et al.*, United States Patent No. 5,627,894.
3. Whether claims 5 and 39 are unpatentable within the meaning of 35 USC §103(b), as being unpatentable over *Hirsch*, in view of *Thomlinson et al.*, United States Patent No. 5,778,069.
4. Whether claims 6, 7, 9-11, 13-16, 40, 41, 43-45, and 47-50 are unpatentable within the meaning of 35 USC §103(b), as being unpatentable over *Hirsch*, in view of *Ming et al.*, United States Patent No. 5,710,815.

5. Whether claims 8 and 42 are unpatentable within the meaning of 35 USC §103(b), as being unpatentable over *Hirsch*, in view of *Ming et al.*, and further in view of *Anshel et al.*, U.S. Patent No. 5,751,808.

6. Whether claims 12 and 46 are unpatentable within the meaning of 35 USC §103(b), as being unpatentable over *Hirsch*, in view of *Ming et al.*, and further in view of *Albert et al.*, U.S. Patent No. 5,627,738.

7. Whether claims 17 and 51 are unpatentable within the meaning of 35 USC §103(b), as being unpatentable over *Hirsch*, in view of *Ming et al.*, and further in view of *Anshel et al.*, U.S. Patent No. 5,751,808.

8. Whether claims 18, 20-24, 52, and 54-58 are unpatentable within the meaning of 35 USC §103(b), as being unpatentable over *Hirsch*, in view of *Anshel et al.*

9. Whether claims 19 and 53 are unpatentable within the meaning of 35 USC §103(b), as being unpatentable over *Hirsch*, in view of *Anshel et al.*, and further in view of *Albert et al.*

10. Whether claims 25, 27-31, 59, and 61-65 are unpatentable within the meaning of 35 USC §103(b), as being unpatentable over *Hirsch*, in view of *Tomko et al.*, United States Patent No. 5,541,994.

11. Whether claims 26 and 60 are unpatentable within the meaning of 35 USC §103(b), as being unpatentable over *Hirsch*, in view of *Tomko et al.*, and further in view of *Albert et al.*

GROUPING OF CLAIMS

With respect to the issues on appeal, the claims are grouped as follows.

Group 1: Claims 1, 2, 4, 32-36, 38, and 67-69. The claims in this group do not stand and fall together.

Group 2: Claims 3 and 37. The claims in this group do not stand and fall together.

Group 3: Claims 5 and 39. The claims in this group do not stand and fall together.

Group 4: Claims 6, 7, 9-11, 13-16, 40, 41, 43-45, and 47-50. The claims in this group do not stand and fall together.

Group 5: Claims 8 and 42. The claims in this group do not stand and fall together.

Group 6: Claims 12 and 46. The claims in this group do not stand and fall together.

Group 7: Claims 17 and 51. The claims in this group do not stand and fall together.

Group 8: Claims 18, 20-24, 52, and 54-58. The claims in this group do not stand and fall together.

Group 9: Claims 19 and 53. The claims in this group do not stand and fall together.

Group 10: Claims 25, 27-31, 59, and 61-65. The claims in this group do not stand and fall together.

Group 11: Claims 26 and 60. The claims in this group do not stand and fall together.

FIRST ARGUMENT

The Examiner's rejection of claims 1, 2, 4, 32-36, 38, and 67-69 as being anticipated by *Hirsch* is not supported by the *Hirsch* disclosure. *Hirsch* does not disclose the cryptographic key split combiner, process for forming cryptographic keys, or cryptographic key recited in the claims, which all require a plurality of key splits that are generated from seed data and randomized to form the key. Rather, *Hirsch* discloses a system for generating a key from a single input value. This system has a random element for modifying the input value, and generates a key, but otherwise bears no resemblance to the claimed invention.

A. THE EXAMINER'S POSITION

The Examiner asserted that *Hirsch* illustrates a cryptographic key split combiner, a "process for combining", and a key formed by the process, corresponding to the invention as recited in claims 1, 35, and 66. The Examiner asserted that *Hirsch* discloses a plurality of key split generators for generating cryptographic key splits (citing column 1, lines 57-67); and a key split randomizer for randomizing the cryptographic key splits to produce the cryptographic key (citing column 1, lines 54-57 and 62-68; column 2, lines 1-7; column 3, lines 60-65; and Fig. 1A, elements 10, 12, and 16); in which each of the key split generators includes means for generating key splits from seed data (citing column 1, lines 49-54 and 62-64).

Regarding claims 2 and 36, the Examiner asserted that *Hirsch* further teaches that the plurality of key split generators includes a random split generator for generating a random key split based on reference data, citing column 2, lines 55-58.

Regarding claims 4 and 38, the Examiner asserted that *Hirsch* suggests that the random split generator includes means for generating a pseudorandom sequence based on the reference data, citing column 2, lines 23-29.

Regarding claims 32 and 67, the Examiner asserted that *Hirsch* discusses that the cryptographic key is a stream of symbols, citing column 4, lines 33-44.

Regarding claims 33 and 68, the Examiner asserted that *Hirsch* describes that the key is at least one symbol block, citing column 4, lines 38-40.

Regarding claims 34 and 69, the Examiner asserted that *Hirsch* specifies that the cryptographic key is a key matrix, citing column 2, lines 5-7.

B. APPELLANT'S POSITION

1. Overview of *Hirsch*

Hirsch discloses a software data protection mechanism, which includes an apparatus for generating a key. The description of the key generating apparatus is set forth in detail from column 3, line 47 through column 5, line 28, with reference to Figs. 1A-C. First, a 32-bit binary value is loaded into an input register 12. Each bit of the 32-bit value is provided as an input to a corresponding scrambler array container 140-n, where n ranges from 0 to 31. Each container 140-n in the scrambler array 14 determines whether the input bit of that container is passed to a respective bit position of a 32-bit output register 18, or whether the complement of that bit is passed instead.

The control for determining whether the input bit or its complement is passed is determined by every fifth bit of a pseudorandom sequence that is generated by a

pseudorandom number generator 16. This generator takes a single seed and generates a single sequence that is loaded serially into the containers of the scrambler array 14, such that five bits are stored in each container. Because there are 32 containers, a 5-bit number also designates each container (bit position) (00000, 00001, 00010, 00011, ... , 11101, 11110, 11111). The result of an XOR operation on the LSB of the bit position and the LSB of the 5-bit pseudorandom sequence segment stored in the container determines whether the input bit or its complement will be provided to the output register. Because the LSB of the bit position alternates in the sequence of scrambler array containers, the XOR result for all even-numbered containers will be the pseudorandom number LSB, and the XOR result for all odd-numbered containers will be the pseudorandom number LSB complement. Thus, the XOR result, and therefore the determination of whether the input bit or its complement is provided to the output register, is determined solely by the pseudorandom sequence. The actual bit position of the output register to which the respective "scrambled" input value bits is provided is determined by the actual value of the pseudorandom number, although the particular correspondence is not disclosed by *Hirsch*. Thus, the original 32-bit value is modified to provide another 32-bit value.

The 32-bit value now stored in the output register 18 is encoded by mapping the values of grouped bits according to a table. That is, the 32 bits of output data are separated into four groups of eight bits each. Each 8-bit group is stored in a set of two overlapping 5-bit registers 200-1a (bits 0-4) and 200-1b (bits 4-8). The outputs of these registers (eight 5-bit values) are provided to a fixed alphanumeric table 204, which maps the input values to provide eight alphanumeric output values. The sequence consisting of these output values is the key, which is stored in a user key register 24.

2. Distinction between the subject matter of claims 1, 2, 4, 32-36, 38, and 67-69 and the *Hirsch* disclosure

Independent claim 1 recites a cryptographic key split combiner. The combiner includes a plurality of key split generators for generating cryptographic key splits, and a key split randomizer for randomizing the cryptographic key splits to produce a cryptographic key. Each of the key split generators includes means for generating key splits from seed data.

In contrast, *Hirsch* discloses a software data protection mechanism, which includes an apparatus for generating a key. In summary of the overview of *Hirsch* stated above, a single 32-bit binary value is loaded into an input register 12. This value is shifted into an output register 18, with some of the bits complemented and their relative positions changed, as determined by a pseudorandom sequence. The resulting single 32-bit value stored in the output register 18 is encoded by mapping the values of grouped bits according to a fixed table. The output of the fixed mapping function is an 8-bit sequence, which is the key.

Thus, where claim 1 recites a plurality of key split generators, each receiving seed data and generating a respective key split, and a randomizer for receiving and randomizing the plurality of key splits to generate a key, *Hirsch* discloses an array 14 for rearranging and complimenting bits of a single value according to a single pseudorandom sequence generated from a single seed, and an encoder 20 for fixed mapping of the modified value to generate the key. That is, *Hirsch* does not disclose a plurality of key split generators, each receiving seed data and generating a respective key split, as recited in claim 1. Rather, *Hirsch* discloses a plurality of containers in a single scrambler array,

which together receive a single 32-bit value that is modified according to a serially-shifted pseudorandom sequence generated from a single seed. Further, *Hirsch* does not disclose randomizing a plurality of separately generated key splits to generate a key, as recited in claim 1. Rather, *Hirsch* discloses taking a single 32-bit modified value, and mapping 8-bit segments of that value according to a stored, predetermined, fixed table, to generate a key.

In summary, the combiner of claim 1 takes separate key splits, each generated based on seed data, and randomizes the splits to provide a key. In contrast, *Hirsch* takes a single input value, modifies the value according to a pseudorandom stream, and maps the modified value according to a fixed table to generate the key. If the modification of the initial value by the scrambler array 14 is considered to be analogous to the randomizer of claim 1 because it is the only pseudorandom function disclosed by *Hirsch* (erroneously, because a key is not produced as a result), then *Hirsch* discloses no key splits, just a single input value, and accordingly no key split generators working from seed data. If the outputs of the overlapping registers 200-1a, 200-1b are considered to be analogous to the key splits of claim 1 (erroneously, because they are not provided by a plurality of key split generators, each operating on seed data), then *Hirsch* discloses no randomizer because the mapping of the register 18 outputs is fixed by the table 204. It is clear that the *Hirsch* apparatus is quite different from the combiner recited in claim 1, and does not include any of the elements recited in claim 1: no key split generators, no key splits, and no key split randomizer. Thus, *Hirsch* cannot anticipate the invention recited in claim 1, nor that recited in claims 2, 4, and 32-34, which all depend from claim 1. The rejection of these claims, therefore, should be withdrawn.

Claim 35 recites a process for forming cryptographic keys. The process includes generating a plurality of cryptographic key splits from seed data; and randomizing the cryptographic key splits to produce a cryptographic key.

As noted above, *Hirsch* does not disclose this process. *Hirsch* does not generate a plurality of cryptographic key splits; *Hirsch* generates a single modified value from a single 32-bit value. *Hirsch* does not generate key splits from seed data. *Hirsch* only generates a pseudorandom sequence from a seed; the sequence is used to control modification of the input value, not to generate a split on which the key is based, as recited in claim 1. *Hirsch* does not disclose randomizing a plurality of key splits to produce a cryptographic key; *Hirsch* discloses mapping a single modified value according to a fixed table to generate a key. The only random activity disclosed by *Hirsch* is control for the selection of input bits to be complemented and rearranged prior to mapping. This random activity is not performed on key splits generated from seed data, and does not produce a cryptographic key. The *Hirsch* key is produced by a fixed mapping table.

Thus, *Hirsch* does not disclose the process recited in claim 35, nor that recited in claims 36 and 38, which depend from claim 35. The rejection of these claims, therefore, should be withdrawn.

Claim 66 recites a cryptographic key formed by the process of claim 35. As discussed above, *Hirsch* discloses a different process, and a different apparatus for carrying out the key-generating process, and therefore cannot disclose the resulting recited key. *Hirsch*, therefore, cannot anticipate the invention recited in claim 66, nor that recited in claims 67-69, which depend from claim 66. The rejection of these claims, therefore, should be withdrawn.

SECOND ARGUMENT

The Examiner's rejection of claims 3 and 37 as being unpatentable over *Hirsch*, in view of *Albert et al.*, is not supported by the teachings of the references. First, it would be improper to attempt to combine the teachings of the cited references, which do not disclose or suggest any motivation for such combination, or even how any such modification could be accomplished. Further, even if such combination were possible and proper, any resulting system still would not include all the elements of the claimed invention.

A. THE EXAMINER'S POSITION

The Examiner asserted that *Hirsch* teaches the claimed cryptographic key split combiner and process, and describes that the random key split generator includes means for generating a pseudorandom sequence based on reference data (citing column 2, lines 23-29), but doesn't explicitly mention generating a random sequence. The Examiner asserted that *Albert et al.* "specify" generating a random sequence (citing column 1, lines 51-67 and column 2, lines 1 and 2). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and "process of combining" of *Hirsch* with generating a random sequence of *Albert et al.* to "increase the quality of random numbers with respect to their predictability and their functional link" (citing column 1, lines 66 and 67, and column 2, lines 1 and 2). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

B. APPELLANT'S POSITION

1. Overview of *Hirsch*

Hirsch discloses a software data protection mechanism, which includes an apparatus for generating a key. The description of the key generating apparatus is set forth in detail from column 3, line 47 through column 5, line 28, with reference to Figs. 1A-C. First, a 32-bit binary value is loaded into an input register 12. Each bit of the 32-bit value is provided as an input to a corresponding scrambler array container 140-n, where n ranges from 0 to 31. Each container 140-n in the scrambler array 14 determines whether the input bit of that container is passed to a respective bit position of a 32-bit output register 18, or whether the complement of that bit is passed instead.

The control for determining whether the input bit or its complement is passed is determined by every fifth bit of a pseudorandom sequence that is generated by a pseudorandom number generator 16. This generator takes a single seed and generates a single sequence that is loaded serially into the containers of the scrambler array 14, such that five bits are stored in each container. Because there are 32 containers, a 5-bit number also designates each container (bit position) (00000, 00001, 00010, 00011, ... , 11101, 11110, 11111). The result of an XOR operation on the LSB of the bit position and the LSB of the 5-bit pseudorandom sequence segment stored in the container determines whether the input bit or its complement will be provided to the output register. Because the LSB of the bit position alternates in the sequence of scrambler array containers, the XOR result for all even-numbered containers will be the pseudorandom number LSB, and

the XOR result for all odd-numbered containers will be the pseudorandom number LSB complement. Thus, the XOR result, and therefore the determination of whether the input bit or its complement is provided to the output register, is determined solely by the pseudorandom sequence. The actual bit position of the output register to which the respective "scrambled" input value bits is provided is determined by the actual value of the pseudorandom number, although the particular correspondence is not disclosed by *Hirsch*. Thus, the original 32-bit value is modified to provide another 32-bit value.

The 32-bit value now stored in the output register 18 is encoded by mapping the values of grouped bits according to a table. That is, the 32 bits of output data are separated into four groups of eight bits each. Each 8-bit group is stored in a set of two overlapping 5-bit registers 200-1a (bits 0-4) and 200-1b (bits 4-8). The outputs of these registers (eight 5-bit values) are provided to a fixed alphanumeric table 204, which maps the input values to provide eight alphanumeric output values. The sequence consisting of these output values is the key, which is stored in a user key register 24.

2. Overview of *Albert et al.*

Albert et al. merely disclose a random number generator, particularly for use in a cryptographic system.

3. Distinction between the subject matter of claims 3 and 37 and the *Hirsch* and *Albert et al.* disclosures

Claim 3 depends from claim 2, which in turn depends from claim 1. Claim 3 thus recites a cryptographic key split combiner. The combiner includes a plurality of key split generators for generating cryptographic key splits, and a key split randomizer for

randomizing the cryptographic key splits to produce a cryptographic key. Each of the key split generators includes means for generating key splits from seed data. The plurality of key split generators includes a random split generator for generating a random key split based on reference data. The random split generator includes means for generating a random sequence based on the reference data.

As noted above, *Hirsch* does not disclose the combiner recited in claim 1. *Albert et al.* merely disclose a random number generator. There is no suggestion by *Albert et al.* that the disclosed random number generator should or could be used as part of a random split generator in a cryptographic key split combiner as recited in claim 1. Further, *Albert et al.* does not overcome the deficiencies of *Hirsch* in disclosing the combiner recited in claim 1. That is, *Hirsch* is deficient in not disclosing a combiner that takes key splits generated by a plurality of key split generators, based on seed data, and randomizes the splits to provide a key, as recited in claim 1. *Hirsch* modifies a single input value according to a pseudorandom sequence, and maps the modified value according to a fixed table to produce a key. *Albert et al.*, by merely describing a random number generator, do not provide *Hirsch* with a means for generating a plurality of key splits from seed data, or for randomizing the plurality of splits to provide a key.

The Examiner does not specify how the teachings of these two references could be combined, but implied that the *Albert et al.* random number generator could be substituted for the *Hirsch* pseudorandom number generator to provide less predictability in the generated sequence. However, the *Hirsch* pseudorandom number generator does not provide a key split, that is, a component of the generated key. Rather, this generator only provides a control element used to modify the single 32-bit input in the scrambler

array, which is the only basis of the generated key. The pseudorandom sequence is a functional input, rather than a key component. Modifying the *Hirsch* system to accept the *Albert et al.* generator would merely provide the *Hirsch* system with random scrambler control rather than pseudorandom scrambler control. Neither reference discloses or suggests how or even if the *Hirsch* system could be modified to accept a random output from the *Albert et al.* generator as a key split, how this would be randomized with another key component, or why any such modification would be advantageous.

For at least all the above-noted reasons, no combination of the teachings of the cited references could render obvious the invention recited in claim 3. The rejection of claim 3, therefore, should be withdrawn.

Claim 37 depends from claim 36, which in turn depends from claim 35. Claim 37 thus recites a process for forming cryptographic keys. The process includes generating a plurality of cryptographic key splits from seed data; and randomizing the cryptographic key splits to produce a cryptographic key. Generating a plurality of cryptographic key splits includes generating a random key split based on reference data. Generating a random key split includes generating a random sequence based on the reference data

As noted above, *Hirsch* does not disclose the process recited in claim 35. *Albert et al.* merely disclose a random number generator. There is no suggestion by *Albert et al.* that the disclosed random number generator should or could be used to generate a random key split in the process recited in claim 35. Further, *Albert et al.* does not overcome the deficiencies of *Hirsch* in disclosing the process recited in claim 35. That is, *Hirsch* is deficient in not disclosing generation of a plurality of cryptographic key splits, generation of key splits from seed data, or randomization of key splits to produce a cryptographic

key, as recited in claim 35. *Hirsch* generates a single modified value from a single 32-bit input value, generates a pseudorandom sequence used only to control modification of the input value rather than to generate a split on which the key is based, and maps the single modified value to generate a key. *Albert et al.*, by merely describing a random number generator, do not provide *Hirsch* with a means for generating a plurality of key splits from seed data, or for randomizing the plurality of splits to provide a key, as recited in claim 35.

The Examiner does not specify how the teachings of these two references could be combined, but implied that the *Albert et al.* random number generator could be substituted for the *Hirsch* pseudorandom number generator to provide less predictability in the generated sequence. However, the *Hirsch* pseudorandom number generator does not provide a key split, that is, a component of the generated key. Rather, this generator only provides a control element used to modify the single 32-bit input in the scrambler array, which is the only basis of the generated key. The pseudorandom sequence is a functional input, rather than a key component. Modifying the *Hirsch* system to accept the *Albert et al.* generator would merely provide the *Hirsch* system with random scrambler control rather than pseudorandom scrambler control. Neither reference discloses or suggests how or even if the *Hirsch* system could be modified to accept a random input from the *Albert et al.* generator as a key split, how this would be randomized with another key component, or why any such modification would be advantageous.

For at least all the above-noted reasons, no combination of the teachings of the cited references could render obvious the invention recited in claim 37. The rejection of claim 37, therefore, should be withdrawn.

THIRD ARGUMENT

The Examiner's rejection of claims 5 and 39 as being unpatentable over *Hirsch*, in view of *Thomlinson et al.*, is not supported by the teachings of the references. First, it would be improper to attempt to combine the teachings of the cited references, which do not disclose or suggest any motivation for such combination, or even how any such modification could be accomplished. Further, even if such combination were possible and proper, any resulting system still would not include all the elements of the claimed invention.

A. THE EXAMINER'S POSITION

The Examiner asserted that *Hirsch* teaches the claimed cryptographic key split combiner and process, but does not explicitly show chronological data. The Examiner asserted that *Thomlinson et al.* disclose generating a key split based on reference data and on chronological data (citing column 3, lines 16-23). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and "process of combining" of *Hirsch* with generating a key split based on chronological data of *Thomlinson et al.* "to ensure unguessability" (citing column 3, lines 2-7). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

B. APPELLANT'S POSITION

1. Overview of Hirsch

Hirsch discloses a software data protection mechanism, which includes an apparatus for generating a key. The description of the key generating apparatus is set forth in detail from column 3, line 47 through column 5, line 28, with reference to Figs. 1A-C. First, a 32-bit binary value is loaded into an input register 12. Each bit of the 32-bit value is provided as an input to a corresponding scrambler array container 140-n, where n ranges from 0 to 31. Each container 140-n in the scrambler array 14 determines whether the input bit of that container is passed to a respective bit position of a 32-bit output register 18, or whether the complement of that bit is passed instead.

The control for determining whether the input bit or its complement is passed is determined by every fifth bit of a pseudorandom sequence that is generated by a pseudorandom number generator 16. This generator takes a single seed and generates a single sequence that is loaded serially into the containers of the scrambler array 14, such that five bits are stored in each container. Because there are 32 containers, a 5-bit number also designates each container (bit position) (00000, 00001, 00010, 00011, ... , 11101, 11110, 11111). The result of an XOR operation on the LSB of the bit position and the LSB of the 5-bit pseudorandom sequence segment stored in the container determines whether the input bit or its complement will be provided to the output register. Because the LSB of the bit position alternates in the sequence of scrambler array containers, the XOR result for all even-numbered containers will be the pseudorandom number LSB, and the XOR result for all odd-numbered containers will be the pseudorandom number LSB

complement. Thus, the XOR result, and therefore the determination of whether the input bit or its complement is provided to the output register, is determined solely by the pseudorandom sequence. The actual bit position of the output register to which the respective “scrambled” input value bits is provided is determined by the actual value of the pseudorandom number, although the particular correspondence is not disclosed by *Hirsch*. Thus, the original 32-bit value is modified to provide another 32-bit value.

The 32-bit value now stored in the output register 18 is encoded by mapping the values of grouped bits according to a table. That is, the 32 bits of output data are separated into four groups of eight bits each. Each 8-bit group is stored in a set of two overlapping 5-bit registers 200-1a (bits 0-4) and 200-1b (bits 4-8). The outputs of these registers (eight 5-bit values) are provided to a fixed alphanumeric table 204, which maps the input values to provide eight alphanumeric output values. The sequence consisting of these output values is the key, which is stored in a user key register 24.

2. Overview of *Thomlinson et al.*

Thomlinson et al. disclose a non-biased pseudorandom number generator. The generator includes an input device that gathers classes of bits provided elsewhere in a computer to hash a seed used to generate the pseudorandom number. One of the classes of bits is a machine class that relates to operating parameters of the computer, such as time of day and date.

3. Distinction between the subject matter of claims 5 and 39 and the *Hirsch* and *Thomlinson et al.* disclosures

Claim 5 depends from claim 2, which in turn depends from claim 1. Thus, claim 5 recites a cryptographic key split combiner. The combiner includes a plurality of key split generators for generating cryptographic key splits, and a key split randomizer for randomizing the cryptographic key splits to produce a cryptographic key. Each of the key split generators includes means for generating key splits from seed data. The plurality of key split generators includes a random split generator for generating a random key split based on reference data. The random split generator includes means for generating a key split based on the reference data and on chronological data.

As discussed previously, *Hirsch* does not disclose the combiner recited in claim 1. *Thomlinson et al.* merely disclose a pseudorandom number generator, which operates on a seed that is based in part on machine bits that may include chronological data. There is no suggestion by *Thomlinson et al.* that the disclosed number generator should or could be used as part of a random split generator in a cryptographic key split combiner as recited in claim 5. Further, the *Thomlinson et al.* reference does not overcome the deficiencies of *Hirsch* in disclosing the combiner recited in claim 1. That is, *Hirsch* is deficient in not disclosing a combiner that takes a plurality of key splits, each generated based on seed data, and randomizes the plurality of splits to provide a key, as recited in claim 1. *Thomlinson et al.*, by merely describing a pseudorandom number generator, albeit one that may operate on chronological data, do not provide *Hirsch* with a means for generating a plurality of key splits from seed data, or for randomizing the plurality of splits to provide a key.

The Examiner does not specify how the teachings of these two references could be combined, but implied that the *Thomlinson et al.* pseudorandom number generator could

be substituted for the *Hirsch* pseudorandom number generator, because the time and date component of the number would make the number harder to guess. Neither reference provides any basis for this contention, and the Examiner does not give any reason for his conclusion. In fact, neither reference provides any motivation for combining those teachings. If the Examiner is instead asserting that the *Thomlinson et al.* pseudorandom sequence could be used as a key split in addition to the input value disclosed by *Hirsch*, he is using hindsight, which is improper. *Thomlinson et al.* do not suggest using the generator output as a key split, and *Hirsch* makes no provision whatsoever for accepting an additional input value.

Regardless of whether such combination is suggested, the combined teachings of the references still would not disclose the invention recited in claim 5. The *Hirsch* pseudorandom number generator does not provide a key split, that is, a component of the generated key. Rather, this generator provides one control factor used to modify the single 32-bit input in the scrambler array, which is the only component of the generated key. It is a functional input, rather than a key component. Thus, substituting the *Thomlinson et al.* pseudorandom number for the *Hirsch* pseudorandom number is simply replacing one number that is not a key split for another number that is not a key split. Further, if the *Thomlinson et al.* pseudorandom sequence were somehow provided to the *Hirsch* system as a split in addition to the modified input value, these splits would presumably be combined by the fixed mapping table, and would not be randomized. As a result, the limitations of claims 5 are still not satisfied, even if such combination of teachings were proper and somehow implemented.

For at least all the above-noted reasons, no combination of the teachings of the cited references could render obvious the invention recited in claim 5. The rejection of claim 5, therefore, should be withdrawn.

Claim 39 depends from claim 36, which in turn depends from claim 35. Thus, claim 39 recites a process for forming cryptographic keys. The process includes generating a plurality of cryptographic key splits from seed data; and randomizing the cryptographic key splits to produce a cryptographic key. Generating a plurality of cryptographic key splits includes generating a random key split based on reference data. Generating a random key split includes generating a key split based on the reference data and on chronological data.

As noted previously, *Hirsch* does not disclose the process recited in claim 35. *Thomlinson et al.* merely disclose a pseudorandom number generator, which operates on a seed that is based in part on machine bits that may include chronological data. There is no suggestion by *Thomlinson et al.* that the disclosed random number generator should or could be used to generate a random key split in the process recited in claim 39. Further, the *Thomlinson et al.* reference does not overcome the deficiencies of *Hirsch* in disclosing the process recited in claim 35. That is, *Hirsch* is deficient in not disclosing generation of a plurality of cryptographic key splits, generation of key splits from seed data, or randomization of the plurality of key splits to produce a cryptographic key, as recited in claim 35. *Hirsch* generates a single modified value from a single 32-bit input value, generates a pseudorandom sequence used only to modify the input value rather than to generate a split on which the key is based, and maps the single modified value to generate a key. *Thomlinson et al.*, by merely describing a pseudorandom number

generator, albeit one that may operate on chronological data, do not provide *Hirsch* with a means for generating a plurality of key splits from seed data, or for randomizing the plurality of splits to provide a key, as recited in claim 35.

The Examiner does not specify how the teachings of these two references could be combined, but implied that the *Thomlinson et al.* pseudorandom number generator could be substituted for the *Hirsch* pseudorandom number generator, because the time and date component of the number would make the number harder to guess. Neither reference provides any basis for this contention, and the Examiner does not give any reason for his conclusion. In fact, neither reference provides any motivation for combining these teachings. If the Examiner is instead asserting that the *Thomlinson et al.* pseudorandom sequence could be used as a key split in addition to the input value disclosed by *Hirsch*, he is using hindsight, which is improper. *Thomlinson et al.* do not suggest using the generator output as a key split, and *Hirsch* makes no provision whatsoever for accepting an additional input value.

Regardless of whether such combination is suggested, the combined teachings of the references still would not disclose the invention recited in claim 39. The *Hirsch* pseudorandom number generator does not provide a key split, that is, a component of the generated key. Rather, this generator provides the sole control factor used to modify the single 32-bit input in the scrambler array, which is the only component of the generated key. It is a functional input, rather than a key component. Thus, substituting the *Thomlinson et al.* pseudorandom number for the *Hirsch* pseudorandom number is simply replacing one number that is not a key split for another number that is not a key split. Further, if the *Thomlinson et al.* pseudorandom sequence were somehow provided to the

Hirsch system as a split in addition to the modified input value, these splits would presumably be combined by the fixed mapping table, and would not be randomized. As a result, the limitations of claims 39 are still not satisfied, even if such combination of teachings would be proper and somehow implemented.

For at least all the above-noted reasons, no combination of the teachings of the cited references could render obvious the invention recited in claim 39. The rejection of claim 39, therefore, should be withdrawn.

FOURTH ARGUMENT

The Examiner's rejection of claims 6, 7, 9-11, 13-16, 40, 41, 43-45, and 47-50 as being unpatentable over *Hirsch*, in view of *Ming et al.*, is not supported by the teachings of the references. First, it would be improper to attempt to combine the teachings of the cited references, which do not disclose or suggest any motivation for such combination, or even how any such modification could be accomplished. Further, even if such combination were possible and proper, any resulting system still would not include all the elements of the claimed invention.

A. THE EXAMINER'S POSITION

Regarding claims 6 and 40, the Examiner asserted that *Hirsch* explains the claimed cryptographic key split combiner and process, but does not explicitly delineate static data. The Examiner asserted that *Ming et al.* discuss generating a key split based on reference data and on static data (citing column 4, lines 4-7). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the

cryptographic key split combiner and process of combining of *Hirsch* with generating a key split based on static data of *Ming et al.* for “implementation of viewer access restrictions” (citing column 7, lines 3-10). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 7 and 41, the Examiner asserted that *Ming et al.* further disclose a means of updating the static data (citing column 4, line 8). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with updating static data of *Ming et al.* for “synchronizing a first pseudo-random number generator within a transmitting unit and a second pseudo-random number generator within a receiving unit” (citing column 3, lines 65-67, and column 4, lines 1-4). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 9 and 43, the Examiner asserted that *Ming et al.* discuss a token split generator for generating a token key split based on label data (citing column 6, lines 26-29; column 5, lines 65-67; column 6, lines 1-5; and column 10, lines 27-34). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with the token split generator of *Ming et al.* for “implementation of viewer access restrictions” (citing column 7, lines 3-10). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 10 and 44, the Examiner asserted that *Ming et al.* “suggest” reading the label data from a storage medium (citing column 7, lines 11-22). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with reading the label data from a storage medium of *Ming et al.* for “implementation of viewer access restrictions” (citing column 7, lines 3-10). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 11 and 45, the Examiner asserted that *Ming et al.* describe that the label data includes user authorization data (citing column 7, lines 11-22). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with user authorization data as label data of *Ming et al.* for “implementation of viewer access restrictions” (citing column 7, lines 3-10). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 13 and 47, the Examiner asserted that *Ming et al.* illustrate a means for generating a pseudorandom sequence based on label data (citing column 13, lines 45-50; Fig. 2, elements 113-115; column 14, lines 39-44; and column 15, lines 40-60). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with the means for generating a pseudorandom sequence based on label data of *Ming et al.* for “implementation of viewer access restrictions” (citing column 7, lines 3-

10). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 14 and 48, the Examiner asserted that *Ming et al.* “specify” the means for generating a key split based on label data and on organization data (citing column 6, lines 26-29 and 59-65; column 5, lines 65-67; and column 6, lines 1-5). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with the means for generating a key split based on label data and on organization data of *Ming et al.* for “implementation of viewer access restrictions” (citing column 7, lines 3-10). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 15 and 49, the Examiner asserted that *Ming et al.* “suggest” a means for generating a key split based on the label data and on static data (citing column 4, lines 4-7). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with the means for generating a key split based on the label data and on static data of *Ming et al.* for “implementation of viewer access restrictions” (citing column 7, lines 3-10). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 16 and 50, the Examiner asserted that *Ming et al.* describe a means for updating the static data (no cite provided). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with the means for updating the

static data of *Ming et al.* for “synchronizing a first pseudo-random number generator within a transmitting unit and a second pseudo-random number generator within a receiving unit” (citing column 3, lines 65-67 and column 4, lines 1-4). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

B. APPELLANT’S POSITION

1. Overview of Hirsch

Hirsch discloses a software data protection mechanism, which includes an apparatus for generating a key. The description of the key generating apparatus is set forth in detail from column 3, line 47 through column 5, line 28, with reference to Figs. 1A-C. First, a 32-bit binary value is loaded into an input register 12. Each bit of the 32-bit value is provided as an input to a corresponding scrambler array container 140-n, where n ranges from 0 to 31. Each container 140-n in the scrambler array 14 determines whether the input bit of that container is passed to a respective bit position of a 32-bit output register 18, or whether the complement of that bit is passed instead.

The control for determining whether the input bit or its complement is passed is determined by every fifth bit of a pseudorandom sequence that is generated by a pseudorandom number generator 16. This generator takes a single seed and generates a single sequence that is loaded serially into the containers of the scrambler array 14, such that five bits are stored in each container. Because there are 32 containers, a 5-bit number also designates each container (bit position) (00000, 00001, 00010, 00011, ... , 11101,

11110, 11111). The result of an XOR operation on the LSB of the bit position and the LSB of the 5-bit pseudorandom sequence segment stored in the container determines whether the input bit or its complement will be provided to the output register. Because the LSB of the bit position alternates in the sequence of scrambler array containers, the XOR result for all even-numbered containers will be the pseudorandom number LSB, and the XOR result for all odd-numbered containers will be the pseudorandom number LSB complement. Thus, the XOR result, and therefore the determination of whether the input bit or its complement is provided to the output register, is determined solely by the pseudorandom sequence. The actual bit position of the output register to which the respective "scrambled" input value bits is provided is determined by the actual value of the pseudorandom number, although the particular correspondence is not disclosed by *Hirsch*. Thus, the original 32-bit value is modified to provide another 32-bit value.

The 32-bit value now stored in the output register 18 is encoded by mapping the values of grouped bits according to a table. That is, the 32 bits of output data are separated into four groups of eight bits each. Each 8-bit group is stored in a set of two overlapping 5-bit registers 200-1a (bits 0-4) and 200-1b (bits 4-8). The outputs of these registers (eight 5-bit values) are provided to a fixed alphanumeric table 204, which maps the input values to provide eight alphanumeric output values. The sequence consisting of these output values is the key, which is stored in a user key register 24.

2. Overview of *Ming et al.*

Ming et al. disclose encoder and decoder apparatus for a cable television signal having embedded viewer access control data. The *Ming et al.* invention provides a means

for scrambling a television signal, and only allowing descrambling of the signal according to selected television programming classes, based on access privileges of a viewer at a television receiver having a decoder unit. Access codes are encrypted and embedded in the transmitted video signal itself, and are read by the decoding unit, which implements the descrambling function if authorized. Programs that the viewer is not authorized to access remain scrambled, wherein the scrambling function takes the form of random line inversion.

3. Distinction between the subject matter of claims 6, 7, 9-11, 13-16, 40, 41, 43-45, and 47-50 and the *Hirsch* and *Ming et al.* disclosures

Claims 6, 7, 9-11, and 13-16 all depend from claim 1, and claims 40, 41, 43-45, and 47-50 all depend from claim 35. As thoroughly discussed above, *Hirsch* does not disclose the apparatus recited in claim 1 and the process recited in claim 35. The *Ming et al.* invention as described above does not overcome the deficiencies of *Hirsch*, that is, *Ming et al.* do not disclose a key split combiner, and do not provide an apparatus adaptable to provide the *Hirsch* system with a plurality of key split generators for producing key splits to be randomized to form a key. *Ming et al.* merely scramble a television signal, and descramble the signal based on a code embedded in the signal. This is reason enough to conclude that no combination of the teachings of *Hirsch* and *Ming et al.* could render obvious the invention recited in claims 6, 7, 9-11, 13-16, 40, 41, 43-45, and 47-50. However, the features recited in these dependent claims even further distinguish the claimed invention from the cited references.

Claim 6 recites a cryptographic key split combiner. The combiner includes a plurality of key split generators for generating cryptographic key splits, and a key split

randomizer for randomizing the cryptographic key splits to produce a cryptographic key. Each of the key split generators includes means for generating key splits from seed data. The plurality of key split generators includes a random split generator for generating a random key split based on reference data. The random split generator includes means for generating a key split based on the reference data and on static data.

Claim 40 recites a process for forming cryptographic keys. The process includes generating a plurality of cryptographic key splits from seed data; and randomizing the cryptographic key splits to produce a cryptographic key. Generating a plurality of cryptographic key splits includes generating a random key split based on reference data and on static data.

With regard to claims 6 and 40, the Examiner stated that *Ming et al.* disclose generating a key split based on static data, at column 4, lines 4-7. That passage describes generating and storing an initial seed value. This seed value is used in a pseudorandom sequence generator as the random function driving the line inverter to scramble the signal. See column 14, line 30 through column 15, line 15. The seed 121 (Fig. 3) is an 8-bit value, the origin of which is undisclosed. See column 15, lines 49-56. Thus, generating and storing the seed value by *Ming et al.* is not the same as generating a static key split. *Ming et al.* do not generate or use a key; the invention simply performs random line inversion, directly according to the generator output. There is no reason to use a key in the *Ming et al.* system because access information is embedded in the signal itself, and is fixed on the receiver end. The only random variable is used to scramble the signal directly. Even if the pseudorandom sequence were considered to be a key, it has no separate components, that is, no splits. Neither *Hirsch* nor *Ming et al.* discloses a

randomizing key split combiner that generates a cryptographic key from a number of splits, which are generated from separate seeds, and no combination of the teachings of these references is suggested, is possible, or could render obvious the invention recited in claims 6 and 40. The rejection of these claims should be withdrawn.

Claim 7 recites the cryptographic key split combiner of claim 6, further including means for updating the static data. Claim 41 recites the process of claim 40, further including updating the static data.

Regarding claims 7 and 41, the Examiner stated that *Ming et al.* disclose a means of updating the static data, at column 4, line 8. That passage describes a process step of generating a next seed value. No detail is provided regarding the seed value or how it is generated, only that it is generated within the transmitting unit. This seed value is used to generate a new pseudorandom sequence for scrambling the subsequent video frame directly. As noted above in discussing the rejection of claims 6 and 40, this is not an update of static data used in generating a key split; there are no keys and hence no splits. There is no suggestion of how or why this feature would or could be combined with the *Hirsch* system. The rejection of claims 7 and 41 should be withdrawn.

Claim 9 recites a cryptographic key split combiner. The combiner includes a plurality of key split generators for generating cryptographic key splits, and a key split randomizer for randomizing the cryptographic key splits to produce a cryptographic key. Each of the key split generators includes means for generating key splits from seed data. The plurality of key split generators includes a token split generator for generating a token split based on label data.

Claim 43 recites a process for forming cryptographic keys. The process includes generating a plurality of cryptographic key splits from seed data; and randomizing the cryptographic key splits to produce a cryptographic key. Generating a plurality of cryptographic key splits includes generating a token key split based on the label data.

Regarding claims 9 and 43, the Examiner stated that *Ming et al.* disclose a token split generator for generating a token key split based on label data, at column 6, lines 26-29; column 5, lines 65-67; and column 6, lines 1-5. In those passages, *Ming et al.* disclose the use of five categories of viewer access. However, these are not key splits. The access control data described by *Ming et al.* is embedded in the video data; it is not used to generate a key split. That is, the receiver has the capability of allowing access in five categories, which may overlap, giving rise to 32 different access levels. A code embedded in the television signal specifies which access level is allowed. If access to a channel is allowed, the signal is descrambled. If access is not allowed, the incoming scrambled signal is maintained. This access control data determines whether the video data gets descrambled at the receiver; it is not used to perform descrambling. Further, it has no bearing at all on the scrambling, which is automatic and independent of the access control data. Therefore, the access control data cannot be part of the key, that is, a key split. See column 13, lines 16-20. It is clear that no combination of the teachings of the cited references is suggested, is possible, or could render obvious the invention recited in claims 9 and 43. The rejection of these claims should be withdrawn.

Claim 10 recites the cryptographic key split combiner of claim 9, further including means for reading the label data from a storage medium. Claim 44 recites the process of claim 43, further including reading the label data from a storage medium.

Regarding claims 10 and 44, the Examiner stated that *Ming et al.* suggest reading label data from a storage medium, at column 7, lines 11-22. That passage describes that the decoder apparatus has a pre-stored user address that corresponds to a user address in the access control data, and which accompanies a user class code. When user class authorization is first implemented, the code is stored at the receiver having an address matching the address in the control data. The class code is pre-stored at the decoder apparatus to compare against the class code embedded in an incoming signal. If there is a match, the signal is descrambled using a pseudorandom code. Thus, while the class code may be analogous to label data, it is not a component of a key used to perform encryption and decryption. It is merely used to perform a simple compare to determine if descrambling will take place by another means. In other words, it is a control value, not a key, and not a key split. The rejection of claims 10 and 44 should be withdrawn.

Claim 11 recites the cryptographic key split combiner of claim 9, where the label data includes authorization data. Claim 45 recites the process of claim 43, where the label data includes authorization data.

Regarding claims 11 and 45, the Examiner stated that *Ming et al.* describe label data that includes user authorization data, at column 7, lines 22-25. That passage describes that the video data includes a program class code identifying authorized classes of users. However, this program class code is not the basis for key split data; it is merely a control value embedded in the video data, and is not a component of a cryptographic key. It is used to determine whether descrambling should take place at the receiver. It is not even a consideration in scrambling the signal, and is applied to the received signal

whether the signal is later descrambled or not, so it isn't a key or a key component. The rejection of claims 11 and 45 should be withdrawn.

Claim 13 recites the cryptographic key split combiner of claim 9, where the token split generator includes means for generating a pseudorandom sequence based on the label data. Claim 47 recites the process of claim 43, where generating the token split includes generating a pseudorandom sequence based on the label data.

Regarding claims 13 and 47, the Examiner stated that *Ming et al.* illustrate a means for generating a pseudorandom sequence, based on label data, at column 13, lines 45-50; Figure 2, items 113-115; and column 14, lines 39-44. In those passages, *Ming et al.* disclose two separate processes. The column 13 and Figure 2 references describe how access control data from two different channel processors are alternately encrypted using the conventional DES algorithm. Thus, the data that the Examiner refers to as the label data is encrypted, but no pseudorandom sequence is generated. The column 14 passage describes the process taking place in Figure 3, which in part shows details of the data formatter and video scramble control block 118 in Figure 2. This block receives an 8-bit seed value 121 from an unknown source, and a line signal 122. Based on these signals, a pseudorandom sequence is generated by the generator 120. This sequence is used to drive the random line inverter. Thus, what the Examiner considers to be label data is encrypted, and data from a completely different source is used to generate a pseudorandom sequence. The "label data" is embedded in the signal, which is scrambled according to the pseudorandom sequence. Thus, the "label data" and the pseudorandom sequence are completely unrelated, both in origin and in application. Further, neither of these has anything to do with a key or with key splits. The "label data" is encrypted for

secure transmission to the decoder apparatus, and the sequence is used to directly scramble the substantive data of the transmission in a manner that is unrelated to the access control data. Access control data is used to determine whether the signal should be descrambled at the receiver, and is not even involved in the actual scrambling process. The pseudorandom sequence is generated from the seed, and is actually used to scramble the signal. The pseudorandom sequence is not based on the “label data”, as required by claims 13 and 47. Clearly, the requirements of claims 13 and 47 are not satisfied. The rejection of these claims should be withdrawn.

Claim 14 recites the cryptographic key split combiner of claim 9, where the token split generator includes means for generating a key split based on the label data and on organization data. Claim 48 recites the process of claim 43, where generating the token split includes generating a key split based on the label data and on organization data.

Regarding claims 14 and 48, the Examiner stated that *Ming et al.* specify the means for generating a key split based on label data and organization data, at column 6, lines 26-29 and 59-65; column 5, lines 65-67; and column 6, lines 1-5. In those passages, *Ming et al.* disclose the use of 32 levels of viewer access, based on five different categories particular to users and classes of users. However, these are not key splits. The access control data described by *Ming et al.* is itself encrypted and is embedded in the video data; it is not used to generate a key split. See column 13, lines 16-20. This data, after being decrypted at the decoder, is the subject of a direct comparison to determine if access is granted. It is not a part of the signal scrambling function, and is applied to the received signal whether it is descrambled or not. Therefore, the access control data cannot be a component of a cryptographic key. It is clear that no combination of the

teachings of the cited references is suggested, is possible, or could render obvious the invention recited in claims 14 and 48. The rejection of these claims should be withdrawn.

Claim 15 recites the cryptographic key split combiner of claim 9, where the token split generator includes means for generating a key split based on the label data and on static data. Claim 49 recites the process of claim 43, where generating the token split includes generating a key split based on the label data and on static data.

Regarding claims 15 and 49, the Examiner stated that *Ming et al.* suggest a means for generating a key split based on the label data and on static data, at column 4, lines 4-7. That passage describes generating and storing an initial seed value. This seed value is used in a pseudorandom sequence generator as the random function driving the line inverter. See column 14, line 30 through column 15, line 15. There is no mention of label data, but as previously observed, what the Examiner considers to be “label data” is completely unrelated to the seed data and the pseudorandom sequence, which certainly is not based on this “label data”. Thus, generating and storing the seed value by *Ming et al.* is not the same as generating a key split based on label data and static data. Even if the pseudorandom sequence were considered to be a key, it has no components, that is, no splits. Neither *Hirsch* nor *Ming et al.* discloses a randomizing key split combiner that generates a cryptographic key from a plurality of splits, which are generated from seed data, and no combination of the teachings of these references is suggested, is possible, or could render obvious the invention recited in claims 15 and 49. The rejection of these claims should be withdrawn.

Claim 16 recites the cryptographic key split combiner of claim 15, further including means for updating the static data. Claim 50 recites the process of claim 49, further including updating the static data.

Regarding claims 16 and 50, the Examiner stated that *Ming et al.* disclose a means of updating the static data, at column 3, lines 65-67 and column 4, lines 1-4. This passage describes synchronizing pseudorandom sequence generators at the receiver and transmitter, so that the scrambled video signal can be descrambled. This is not an update of static data on which a key split is based as generated by a key split generator. It is not used to generate a key split that is randomized with other key splits to generate a cryptographic key. There is no suggestion in either cited reference that this pseudorandom sequence can be randomized with the *Hirsch* modified input value to form a key, and neither reference discloses randomizing key splits. The rejection of claims 16 and 50 should be withdrawn.

In summary, both the *Hirsch* invention and the *Ming et al.* invention are so different from the claimed invention and each other that no combination of the teachings of these two references in an attempt to result in the claimed invention is suggested or possible, and any such combination still would not result in the claimed invention.

FIFTH ARGUMENT

The Examiner's rejection of claims 8 and 42 as unpatentable over *Hirsch*, in view of *Ming et al.* and *Anshel et al.*, is not supported by the teachings of the references. First, it would be improper to attempt to combine the teachings of the cited references, which do not disclose or suggest any motivation for such combination, or even how any such

modification could be accomplished. Further, even if such combination were possible and proper, any resulting system still would not include all the elements of the claimed invention.

A. THE EXAMINER'S POSITION

The Examiner asserted that *Hirsch* in view of *Ming et al.* teaches the claimed cryptographic key split combiner and process. The Examiner also asserted that *Ming et al.* describe modifying a divisor of the static data (citing column 4, lines 18-20). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* in view of *Ming et al.* with modifying a divisor of the static data of *Ming et al.* for “synchronizing a first pseudo-random number generator within a transmitting unit and a second pseudo-random number generator within a receiving unit” (citing column 3, lines 65-67, and column 4, lines 1-4). The Examiner acknowledged that *Ming et al.* do not specify that this value is a prime divisor. The Examiner asserted that *Anshel et al.* show modifying a prime divisor of the static data (citing column 11, lines 8-25, and Fig. 8, element 71). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and “process of combining” of *Hirsch* in view of *Ming et al.* with modifying a prime divisor of the static data of *Anshel et al.* to “generate a cryptographically secure sequence at high speed” (citing column 1, lines 11 and 12). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

B. APPELLANT'S POSITION

1. Overview of *Hirsch*

Hirsch discloses a software data protection mechanism, which includes an apparatus for generating a key. The description of the key generating apparatus is set forth in detail from column 3, line 47 through column 5, line 28, with reference to Figs. 1A-C. First, a 32-bit binary value is loaded into an input register 12. Each bit of the 32-bit value is provided as an input to a corresponding scrambler array container 140-n, where n ranges from 0 to 31. Each container 140-n in the scrambler array 14 determines whether the input bit of that container is passed to a respective bit position of a 32-bit output register 18, or whether the complement of that bit is passed instead.

The control for determining whether the input bit or its complement is passed is determined by every fifth bit of a pseudorandom sequence that is generated by a pseudorandom number generator 16. This generator takes a single seed and generates a single sequence that is loaded serially into the containers of the scrambler array 14, such that five bits are stored in each container. Because there are 32 containers, a 5-bit number also designates each container (bit position) (00000, 00001, 00010, 00011, ... , 11101, 11110, 11111). The result of an XOR operation on the LSB of the bit position and the LSB of the 5-bit pseudorandom sequence segment stored in the container determines whether the input bit or its complement will be provided to the output register. Because the LSB of the bit position alternates in the sequence of scrambler array containers, the XOR result for all even-numbered containers will be the pseudorandom number LSB, and

the XOR result for all odd-numbered containers will be the pseudorandom number LSB complement. Thus, the XOR result, and therefore the determination of whether the input bit or its complement is provided to the output register, is determined solely by the pseudorandom sequence. The actual bit position of the output register to which the respective "scrambled" input value bits is provided is determined by the actual value of the pseudorandom number, although the particular correspondence is not disclosed by *Hirsch*. Thus, the original 32-bit value is modified to provide another 32-bit value.

The 32-bit value now stored in the output register 18 is encoded by mapping the values of grouped bits according to a table. That is, the 32 bits of output data are separated into four groups of eight bits each. Each 8-bit group is stored in a set of two overlapping 5-bit registers 200-1a (bits 0-4) and 200-1b (bits 4-8). The outputs of these registers (eight 5-bit values) are provided to a fixed alphanumeric table 204, which maps the input values to provide eight alphanumeric output values. The sequence consisting of these output values is the key, which is stored in a user key register 24.

2. Overview of *Ming et al.*

Ming et al. disclose encoder and decoder apparatus for a cable television signal having embedded viewer access control data. The *Ming et al.* invention provides a means for scrambling a television signal, and only allowing descrambling of the signal according to selected television programming classes, based on access privileges of a viewer at a television receiver having a decoder unit. Access codes are encrypted and embedded in the transmitted video signal itself, and are read by the decoding unit, which implements the descrambling function if authorized. Programs that the viewer is not authorized to

access remain scrambled, wherein the scrambling function takes the form of random line inversion.

3. Overview of *Anshel et al.*

Anshel et al. disclose a cryptographically secure sequence generator used for pseudorandom sequence generation, authentication, and public key transfer and encryption. The public keys are pseudorandom sequences based on zeta one-way functions, namely quadratic signatures that are special sequences of Jacobi symbols.

4. Distinction between the subject matter of claims 8 and 42 and the *Hirsch, Ming et al.*, and *Anshel et al.* disclosures

Claim 8 depends from claim 1, through intervening claims 7, 6, and 2. Claim 8 thus recites a cryptographic key split combiner, which includes a plurality of key split generators for generating cryptographic key splits, and a key split randomizer for randomizing the cryptographic key splits to produce a cryptographic key. Each of the key split generators includes means for generating key splits from seed data. The plurality of key split generators includes a random split generator for generating a random key split based on reference data. The random split generator includes means for generating a key split based on the reference data and on static data. The cryptographic key split combiner also means for updating the static data, which includes means for modifying a prime number divisor of the static data..

Claim 42 depends from claim 35, through intervening claims 41, 40, and 36. Claim 42 thus recites a process for forming cryptographic keys, which includes generating a plurality of cryptographic key splits from seed data, and randomizing the cryptographic

key splits to produce a cryptographic key. Generating a plurality of cryptographic key splits includes generating a random key split based on reference data and on static data. The process also includes updating the static data, which includes modifying a prime number divisor of the static data.

The Examiner stated that *Ming et al.*, at column 4, lines 18-20, describe modifying the divisor of static data. However, a close reading of this passage does not reveal the subject matter described by the Examiner. Rather, the passage sets forth a procedure for updating seed values for generating the pseudorandom sequence, based on an incremented frame count. In fact, the feature cited by the Examiner is not disclosed anywhere in the reference by *Ming et al.* *Anshel et al.* disclose a cryptographic sequence generator. In the passage cited by the Examiner, *Anshel et al.* describe use of randomly-selected prime numbers to generate a list of Jacobi symbols and, according to a public key, choosing a subset of the symbols to encrypt a single input bit. The Jacobi symbols and the public key both feature terms having prime number divisors. See column 11, lines 8-53. In contrast, claims 8 and 42 recite updating the static data (a basis for a random key split) by modifying a prime number divisor of the static data.

None of the cited references discloses random key splits. It follows that none of the cited references discloses updating a basis of a random key split, or performing the update by modifying a prime number divisor of the basis. *Anshel et al.* disclose the use of prime number divisors in a Jacobi sequence and in a public key. However, demonstrating that it is known to use prime number divisors in an application related to cryptography is not enough to suggest to one of ordinary skill in the art that modification of a prime number divisor of a basis of a random key split is beneficial in generating a key by

randomizing a number of key splits including the random key split, particularly in view of other references that do not even disclose the use of key splits. It is clear that no combination of the teachings of the cited references is suggested in any of the references, or is even possible given their respective disclosed features. Even if these disclosed features were somehow combined, the resulting amalgamation would not include all the features of the claimed invention, and therefore could possibly render obvious the invention recited in claims 8 and 42. The rejection of these claims, therefore, should be withdrawn.

SIXTH ARGUMENT

The Examiner's rejection of claims 12 and 46 as being unpatentable over *Hirsch*, in view of *Ming et al.* and *Albert et al.*, is not supported by the teachings of the references. First, it would be improper to attempt to combine the teachings of the cited references, which do not disclose or suggest any motivation for such combination, or even how any such modification could be accomplished. Further, even if such combination were possible and proper, any resulting system still would not include all the elements of the claimed invention.

A. THE EXAMINER'S POSITION

The Examiner asserted that *Hirsch* in view of *Ming et al.* teach the claimed cryptographic key split combiner and process. The Examiner further asserted that *Ming et al.* illustrate a means for generating a pseudorandom sequence based on label data (citing column 13, lines 45-50; Fig. 2, elements 113-115; and column 14, lines 39-44).

The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* in view of *Ming et al.* with generating a pseudorandom sequence based on label data of *Ming et al.* for implementation of viewer access restrictions (citing column 7, lines 3-10). The Examiner acknowledged that neither *Hirsch* nor *Ming et al.* specifies that this is a random sequence. The Examiner asserted that *Albert et al.* “elaborate on” generating a random sequence (citing column 1, lines 51-67, and column 2, lines 1 and 2). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and “process of combining” of *Hirsch* in view of *Ming et al.* with generating a random sequence of *Albert et al.* to “increase the quality of random numbers with respect to their predictability and their functional link” (citing column 1, lines 66 and 67, and column 2, lines 1 and 2). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

B. APPELLANT’S POSITION

1. Overview of Hirsch

Hirsch discloses a software data protection mechanism, which includes an apparatus for generating a key. The description of the key generating apparatus is set forth in detail from column 3, line 47 through column 5, line 28, with reference to Figs. 1A-C. First, a 32-bit binary value is loaded into an input register 12. Each bit of the 32-bit value is provided as an input to a corresponding scrambler array container 140-n,

where n ranges from 0 to 31. Each container 140- n in the scrambler array 14 determines whether the input bit of that container is passed to a respective bit position of a 32-bit output register 18, or whether the complement of that bit is passed instead.

The control for determining whether the input bit or its complement is passed is determined by every fifth bit of a pseudorandom sequence that is generated by a pseudorandom number generator 16. This generator takes a single seed and generates a single sequence that is loaded serially into the containers of the scrambler array 14, such that five bits are stored in each container. Because there are 32 containers, a 5-bit number also designates each container (bit position) (00000, 00001, 00010, 00011, ..., 11101, 11110, 11111). The result of an XOR operation on the LSB of the bit position and the LSB of the 5-bit pseudorandom sequence segment stored in the container determines whether the input bit or its complement will be provided to the output register. Because the LSB of the bit position alternates in the sequence of scrambler array containers, the XOR result for all even-numbered containers will be the pseudorandom number LSB, and the XOR result for all odd-numbered containers will be the pseudorandom number LSB complement. Thus, the XOR result, and therefore the determination of whether the input bit or its complement is provided to the output register, is determined solely by the pseudorandom sequence. The actual bit position of the output register to which the respective "scrambled" input value bits is provided is determined by the actual value of the pseudorandom number, although the particular correspondence is not disclosed by *Hirsch*. Thus, the original 32-bit value is modified to provide another 32-bit value.

The 32-bit value now stored in the output register 18 is encoded by mapping the values of grouped bits according to a table. That is, the 32 bits of output data are

separated into four groups of eight bits each. Each 8-bit group is stored in a set of two overlapping 5-bit registers 200-1a (bits 0-4) and 200-1b (bits 4-8). The outputs of these registers (eight 5-bit values) are provided to a fixed alphanumeric table 204, which maps the input values to provide eight alphanumeric output values. The sequence consisting of these output values is the key, which is stored in a user key register 24.

2. Overview of *Ming et al.*

Ming et al. disclose encoder and decoder apparatus for a cable television signal having embedded viewer access control data. The *Ming et al.* invention provides a means for scrambling a television signal, and only allowing descrambling of the signal according to selected television programming classes, based on access privileges of a viewer at a television receiver having a decoder unit. Access codes are encrypted and embedded in the transmitted video signal itself, and are read by the decoding unit, which implements the descrambling function if authorized. Programs that the viewer is not authorized to access remain scrambled, wherein the scrambling function takes the form of random line inversion.

3. Overview of *Albert et al.*

Albert et al. merely disclose a random number generator, particularly for use in a cryptographic system.

4. Distinction between the subject matter of claims 12 and 46 and the *Hirsch*, *Ming et al.*, and *Albert et al.* disclosures

Claim 12 depends from claim 1, through intervening claim 9. Claim 8 thus recites a cryptographic key split combiner, which includes a plurality of key split generators for generating cryptographic key splits, and a key split randomizer for randomizing the cryptographic key splits to produce a cryptographic key. The plurality of key split generators includes a token split generator for generating a token key split based on label data. The token split generator includes means for generating a random sequence based on the label data.

Claim 46 depends from claim 35, through intervening claim 43. Claim 46 thus recites a process for forming cryptographic keys, which includes generating a plurality of cryptographic key splits from seed data, and randomizing the cryptographic key splits to produce a cryptographic key. Generating a plurality of cryptographic key splits includes generating a token key split based on label data, which includes generating a random sequence based on the label data.

As noted above, *Hirsch* does not disclose the invention recited in claims 1 and 35. The Examiner asserted that *Ming et al.* disclose generating a pseudorandom sequence based on label data. However, as noted in discussing the rejection of claims 13 and 47, this is not the case. The access control data ("label data", as identified by the Examiner) is encrypted for transmission to the decoder apparatus, and unrelated seed data is used to generate a pseudorandom sequence to drive a line inverter to scramble the video signal. The pseudorandom sequence is not at all based on the access control data. The encrypted access control data is decrypted at the receiver, but is only used to determine if the signal should be descrambled according to the pseudorandom sequence. Further, *Albert et al.* merely disclose a design for a random number generator. *Albert et al.*, by merely

describing a random number generator, do not provide a connection between the *Ming et al.* access control data and pseudorandom sequence, or provide *Hirsch* with a means for generating a plurality of key splits from seed data, or for randomizing the plurality of splits to provide a key.

The Examiner does not specify how the teachings of these three references could be combined. Further, there is no suggestion in the cited references as to how or why the teachings should be combined, and no rational reason exists for doing so. Any such combination of the elements of the cited inventions still would not result in the invention as recited in claims 12 and 46; it is not certain, based on the teachings of the references and on the Examiner's assertions, how the resulting system would be configured or function. The rejection of these claims should be withdrawn.

SEVENTH ARGUMENT

The Examiner's rejection of claims 17 and 51 as unpatentable over *Hirsch*, in view of *Ming et al.* and *Anshel et al.*, is not supported by the teachings of the references. First, it would be improper to attempt to combine the teachings of the cited references, which do not disclose or suggest any motivation for such combination, or even how any such modification could be accomplished. Further, even if such combination were possible and proper, any resulting system still would not include all the elements of the claimed invention.

A. THE EXAMINER'S POSITION

The Examiner asserted that *Hirsch* in view of *Ming et al.* teach the claimed cryptographic key split combiner and process. The Examiner further asserted that *Ming et al.* describe modifying a divisor of the static data (citing column 4, lines 18-20). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and “process of combining” of *Hirsch* in view of *Ming et al.* with modifying a divisor of the static data of *Ming et al.* for “synchronizing a first pseudo-random number generator within a transmitting unit and a second pseudo-random generator within a receiving unit” (citing column 3, lines 65-67, and column 4, lines 1-4). The Examiner acknowledged that *Ming et al.* do not specify that this value is a prime divisor. The Examiner asserted that *Anshel et al.* show modifying a prime divisor of the static data (citing column 11, lines 8-25, and Fig. 8, element 71). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* in view of *Ming et al.* with modifying a prime divisor of the static data of *Anshel et al.* to “generate a cryptographically secure sequence at high speed” (citing column 1, lines 11 and 12). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

B. APPELLANT'S POSITION

1. Overview of Hirsch

Hirsch discloses a software data protection mechanism, which includes an apparatus for generating a key. The description of the key generating apparatus is set forth in detail from column 3, line 47 through column 5, line 28, with reference to Figs. 1A-C. First, a 32-bit binary value is loaded into an input register 12. Each bit of the 32-bit value is provided as an input to a corresponding scrambler array container 140-n, where n ranges from 0 to 31. Each container 140-n in the scrambler array 14 determines whether the input bit of that container is passed to a respective bit position of a 32-bit output register 18, or whether the complement of that bit is passed instead.

The control for determining whether the input bit or its complement is passed is determined by every fifth bit of a pseudorandom sequence that is generated by a pseudorandom number generator 16. This generator takes a single seed and generates a single sequence that is loaded serially into the containers of the scrambler array 14, such that five bits are stored in each container. Because there are 32 containers, a 5-bit number also designates each container (bit position) (00000, 00001, 00010, 00011, ... , 11101, 11110, 11111). The result of an XOR operation on the LSB of the bit position and the LSB of the 5-bit pseudorandom sequence segment stored in the container determines whether the input bit or its complement will be provided to the output register. Because the LSB of the bit position alternates in the sequence of scrambler array containers, the XOR result for all even-numbered containers will be the pseudorandom number LSB, and the XOR result for all odd-numbered containers will be the pseudorandom number LSB

complement. Thus, the XOR result, and therefore the determination of whether the input bit or its complement is provided to the output register, is determined solely by the pseudorandom sequence. The actual bit position of the output register to which the respective “scrambled” input value bits is provided is determined by the actual value of the pseudorandom number, although the particular correspondence is not disclosed by *Hirsch*. Thus, the original 32-bit value is modified to provide another 32-bit value.

The 32-bit value now stored in the output register 18 is encoded by mapping the values of grouped bits according to a table. That is, the 32 bits of output data are separated into four groups of eight bits each. Each 8-bit group is stored in a set of two overlapping 5-bit registers 200-1a (bits 0-4) and 200-1b (bits 4-8). The outputs of these registers (eight 5-bit values) are provided to a fixed alphanumeric table 204, which maps the input values to provide eight alphanumeric output values. The sequence consisting of these output values is the key, which is stored in a user key register 24.

2. Overview of *Ming et al.*

Ming et al. disclose encoder and decoder apparatus for a cable television signal having embedded viewer access control data. The *Ming et al.* invention provides a means for scrambling a television signal, and only allowing descrambling of the signal according to selected television programming classes, based on access privileges of a viewer at a television receiver having a decoder unit. Access codes are encrypted and embedded in the transmitted video signal itself, and are read by the decoding unit, which implements the descrambling function if authorized. Programs that the viewer is not authorized to

access remain scrambled, wherein the scrambling function takes the form of random line inversion.

3. Overview of *Anshel et al.*

Anshel et al. disclose a cryptographically secure sequence generator used for pseudorandom sequence generation, authentication, and public key transfer and encryption. The public keys are pseudorandom sequences based on zeta one-way functions, namely quadratic signatures that are special sequences of Jacobi symbols.

4. Distinction between the subject matter of claims 17 and 51 and the *Hirsch, Ming et al.*, and *Anshel et al.* disclosures

Claim 17 depends from claim 1, through intervening claims 16, 15, and 9. Claim 17 thus recites a cryptographic key split combiner, which includes a plurality of key split generators for generating cryptographic key splits, and a key split randomizer for randomizing the cryptographic key splits to produce a cryptographic key. The plurality of key split generators includes a token split generator for generating a token key split based on label data. The token split generator includes means for generating a key split based on the label data and on static data. The cryptographic key split combiner further includes means for updating the static data, which includes means for modifying a prime number divisor of the static data.

Claim 51 depends from claim 35, through intervening claims 50, 49, and 43. Claim 51 thus recites a process for forming cryptographic keys, which includes generating a plurality of cryptographic key splits from seed data, and randomizing the cryptographic key splits to produce a cryptographic key. Generating a plurality of cryptographic key

splits includes generating a token key split based on label data and on static data. The process further includes updating the static data, which includes modifying a prime number divisor of the static data.

The Examiner stated that *Ming et al.*, at column 4, lines 18-20, describe modifying the divisor of static data. However, a close reading of this passage does not reveal the subject matter described by the Examiner. Rather, the passage sets forth a procedure for updating seed values for generating the pseudorandom sequence, based on an incremented frame count. *Anshel et al.* disclose a cryptographic sequence generator. In the passage cited by the Examiner, *Anshel et al.* describe use of randomly-selected prime numbers to generate a list of Jacobi symbols and, according to a public key, chooses a subset of the symbols to encrypt a single input bit. The Jacobi symbols and the public key both feature terms having prime number divisors. See column 11, lines 8-53. However, claims 17 and 51 recite updating the static data by modifying a prime number divisor of the static data. The static data is a basis for a random key split.

None of the cited references discloses random key splits. It follows that none of the cited references discloses updating a basis of a random key split, or performing the update by modifying a prime number divisor of the basis. *Anshel et al.* disclose the use of prime number divisors in a Jacobi sequence and in a public key. However, demonstrating that it is known to use prime number divisors in an application related to cryptography is not enough to suggest to one of ordinary skill in the art that modification of a prime number divisor of a basis of a random key split is beneficial in generating a key based on the randomization of a number of key splits, particularly in view of other references that do not even disclose the use of key splits. Further, the “motivation” provided by the

Examiner for combining the teachings of *Hirsch* and *Ming et al.* is that it would enable “synchronizing a first pseudo-random generator within a transmitting unit and a second pseudo-random generator within a receiving unit.” However, this is not a function of a cryptographic key or a key split. It is true that, in most cases, the transmitter and receiver must be synchronized to decrypt properly at the receiver. However, this synchronization necessarily takes place independently of application of a key. Thus, the feature pointed out by the Examiner is not related to a key or key split, and cannot provide the motivation to create the key split recited in the claims. It is clear that no combination of the teachings of the cited references could possibly render obvious the invention recited in claims 17 and 51, and further that any such combination is not suggested in the references and is therefore improper. The rejection of these claims, therefore, should be withdrawn.

EIGHTH ARGUMENT

The Examiner’s rejection of claims 18, 20, 24, 52, and 54-58 as being unpatentable over *Hirsch*, in view of *Anshel et al.*, is not supported by the teachings of the references. First, it would be improper to attempt to combine the teachings of the cited references, which do not disclose or suggest any motivation for such combination, or even how any such modification could be accomplished. Further, even if such combination were possible and proper, any resulting system still would not include all the elements of the claimed invention.

A. THE EXAMINER'S POSITION

Regarding claims 18 and 52, the Examiner asserted that *Hirsch* teaches the claimed cryptographic key split combiner and process. The Examiner acknowledged that *Hirsch* does not describe maintenance data. The Examiner asserted that *Anshel et al.* discuss a console split generator for generating a console key split based on maintenance data (citing column 8, lines 8-15). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with generating a console key split based on maintenance data of *Anshel et al.* for "simple and highly secure authentication" (citing column 8, lines 8 and 9). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 20 and 54, the Examiner asserted that *Anshel et al.* describe a means for generating a pseudorandom sequence based on maintenance data (citing column 8, lines 8-15). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with a means for generating a pseudorandom sequence based on maintenance data of *Anshel et al.* for "simple and highly secure authentication" (citing column 8, lines 8 and 9). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 21 and 55, the Examiner asserted that *Anshel et al.* "specify" generating a key split based on previous maintenance data and on current maintenance data (citing column 8, lines 26 and 27). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner

and process of combining of *Hirsch* with generating a key split based on previous maintenance data and on current maintenance data of *Anshel et al.* for “simple and highly secure authentication” (citing column 8, lines 8 and 9). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 22 and 56, the Examiner asserted that *Anshel et al.* “mention” generating a key split based on the maintenance data and on static data (citing column 8, lines 16-22). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with generating a key split based on maintenance data and on static data of *Anshel et al.* for “simple and highly secure authentication” (citing column 8, lines 8 and 9). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 23 and 57, the Examiner asserted that *Anshel et al.* “delineate” a means for updating the static data (citing column 8, lines 8, 26, and 27). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with updating the static data of *Anshel et al.* to “generate a cryptographically secure sequence at high speed” (citing column 1, lines 11 and 12). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 24 and 58, the Examiner asserted that *Anshel et al.* illustrate that updating the static data includes modifying a prime number divisor of the static data

(citing column 11, lines 8-25, and Fig. 8, element 71). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with modifying a prime number divisor of the static data of *Anshel et al.* to “generate a cryptographically secure sequence at high speed” (citing column 1, lines 11 and 12). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

B. APPELLANT’S POSITION

1. Overview of *Hirsch*

Hirsch discloses a software data protection mechanism, which includes an apparatus for generating a key. The description of the key generating apparatus is set forth in detail from column 3, line 47 through column 5, line 28, with reference to Figs. 1A-C. First, a 32-bit binary value is loaded into an input register 12. Each bit of the 32-bit value is provided as an input to a corresponding scrambler array container 140-n, where n ranges from 0 to 31. Each container 140-n in the scrambler array 14 determines whether the input bit of that container is passed to a respective bit position of a 32-bit output register 18, or whether the complement of that bit is passed instead.

The control for determining whether the input bit or its complement is passed is determined by every fifth bit of a pseudorandom sequence that is generated by a pseudorandom number generator 16. This generator takes a single seed and generates a single sequence that is loaded serially into the containers of the scrambler array 14, such

that five bits are stored in each container. Because there are 32 containers, a 5-bit number also designates each container (bit position) (00000, 00001, 00010, 00011, ..., 11101, 11110, 11111). The result of an XOR operation on the LSB of the bit position and the LSB of the 5-bit pseudorandom sequence segment stored in the container determines whether the input bit or its complement will be provided to the output register. Because the LSB of the bit position alternates in the sequence of scrambler array containers, the XOR result for all even-numbered containers will be the pseudorandom number LSB, and the XOR result for all odd-numbered containers will be the pseudorandom number LSB complement. Thus, the XOR result, and therefore the determination of whether the input bit or its complement is provided to the output register, is determined solely by the pseudorandom sequence. The actual bit position of the output register to which the respective "scrambled" input value bits is provided is determined by the actual value of the pseudorandom number, although the particular correspondence is not disclosed by *Hirsch*. Thus, the original 32-bit value is modified to provide another 32-bit value.

The 32-bit value now stored in the output register 18 is encoded by mapping the values of grouped bits according to a table. That is, the 32 bits of output data are separated into four groups of eight bits each. Each 8-bit group is stored in a set of two overlapping 5-bit registers 200-1a (bits 0-4) and 200-1b (bits 4-8). The outputs of these registers (eight 5-bit values) are provided to a fixed alphanumeric table 204, which maps the input values to provide eight alphanumeric output values. The sequence consisting of these output values is the key, which is stored in a user key register 24.

2. Overview of *Anshel et al.*

Anshel et al. disclose a cryptographically secure sequence generator used for pseudorandom sequence generation, authentication, and public key transfer and encryption. The public keys are pseudorandom sequences based on zeta one-way functions, namely quadratic signatures that are special sequences of Jacobi symbols.

3. Distinction between the subject matter of claims 18, 20-24, 52, and 54-58 and the *Hirsch* and *Anshel et al.* disclosures

Claims 18 and 20-24 depend from claim 1, which requires a cryptographic key split combiner, which includes a plurality of key split generators for generating cryptographic key splits, and a key split randomizer for randomizing the cryptographic key splits to produce a cryptographic key. Claims 52 and 54-58 depend from claim 35, which requires a process for forming cryptographic keys, which includes generating a plurality of cryptographic key splits from seed data, and randomizing the cryptographic key splits to produce a cryptographic key.

Claim 18 recites that the plurality of key split generators includes a console split generator for generating a console key split based on maintenance data. Claim 52 recites that generating the plurality of key splits includes generating a console key split based on maintenance data.

Regarding claims 18 and 52, the Examiner stated that *Anshel et al.* discuss a console split generator for generating a console key split based on maintenance data, citing column 8, lines 8-15. The passage cited by the Examiner actually describes a conventional public key infrastructure arrangement. As disclosed, users of a network are each given a fixed (line 10) private key, and a public key is broadcast to users of the

network. Key splits of any kind are not disclosed. As disclosed, the private key is fixed; this would teach away from having a key component that would depend on maintenance data. In the *Anshel et al.* scheme, this private key remains fixed, while each public key is used only once and then discarded. A new public key is generated, based solely on an incremented state value. There is no seed provided to generate a split from maintenance data, or randomizer for combining a console split with other splits to determine a key, as recited in claims 18 and 52. Rather, the key is determined directly from the state data.

As previously discussed, *Hirsch* does not disclose the elements of claims 1 and 35, from which claims 18 and 52 respectively depend. *Anshel et al.* do not make up for the deficiencies of *Hirsch*, do not disclose the features of the dependent claims, and teach away from any such disclosure or combination. Thus, no combination of the teachings of these references could render obvious the invention recited in claims 18 and 52. The rejection of these claims should be withdrawn.

Claim 20 recites that the console split generator includes means for generating a pseudorandom sequence based on the maintenance data. Claim 54 recites that generating the console key split includes generating a pseudorandom sequence based on the maintenance data.

Regarding claims 20 and 54, the Examiner stated that *Anshel et al.* disclose means for generating a pseudorandom sequence based on maintenance data, citing column 8, lines 8-15. As described in the passage and shown in Figure 4, the ZPNG generates a pseudorandom code that is transformed by a zeta code transformer to produce the public key. This, therefore, has nothing to do with a split generator that generates one of a number of key splits that are randomized to form a cryptographic key. Rather, the

sequence itself is transformed to become the key, with no additional components. As previously discussed, *Hirsch* does not disclose such a combiner, so it would be pointless to apply the teachings of *Anshel et al.* to those of *Hirsch* to show that the *Anshel et al.* key could be a key split in the *Hirsch* system, because the *Hirsch* system does not accommodate splits, and has no randomizer to create a key from splits. Thus, no combination of the teachings of these references could render obvious the invention recited in claims 20 and 54. The rejection of these claims should be withdrawn.

Claim 21 recites that the console split generator includes means for generating a key split based on previous maintenance data and on current maintenance data. Claim 55 recites that generating the console key split includes generating a key split based on previous maintenance data and on current maintenance data.

Regarding claims 21 and 55, the Examiner stated that *Anshel et al.* specify generating a key split based on previous and current maintenance data, citing column 8, lines 26 and 27. As described in the passage, a public key is generated based on a current state value. The key is used once, and discarded. The state value is incremented, and a new public key is determined based on this incremented value. Thus, the public key is determined based only on a current state value. The previous state value and current state value are never both used to determine the public key, as required by claims 21 and 55. See also Figure 4. Further, this key is not a key split, as discussed previously. Also as previously discussed, *Hirsch* does not disclose the claimed combiner. Thus, no combination of the teachings of these references could render obvious the invention recited in claims 21 and 55. The rejection of these claims should be withdrawn.

Claim 22 recites that the console split generator includes means for generating a key split based on the maintenance data and on static data. Claim 56 recites that generating the console key split includes generating a key split based on the maintenance data and on static data.

Regarding claims 22 and 56, the Examiner stated that *Anshel et al.* mention generating a key split based on maintenance data and static data, citing column 8, lines 16-22. The passage cited by the Examiner actually describes generation of the public key of a conventional public key infrastructure arrangement. As disclosed, the public key is generated based on an incremented state value. There is no seed provided to generate a split from maintenance data, or randomizer for combining a console split with other splits to determine a key, as recited in claims 22 and 56. Rather, the key is determined directly from the state data. As previously discussed, *Hirsch* does not disclose the elements of claims 1 and 35, from which claims 22 and 56 respectively depend. That is, like *Anshel et al.*, *Hirsch* does not disclose a number of key splits, generated from different seeds, that are randomized to generate a key. Thus, no combination of the teachings of these references could render obvious the invention recited in claims 22 and 56. The rejection of these claims should be withdrawn.

Claim 23 recites that the combiner of claim 22 includes means for updating the static data. Claim 57 recites that the process of claim 56 includes updating the static data.

Regarding claims 23 and 57, the Examiner stated that *Anshel et al.* delineate a means for updating static data, citing column 8, lines 8, 26, and 27. This passage describes generating a new public key based on an incremented state value. It is believed that the Examiner previously identified this disclosed state data as the claimed

maintenance data. Claims 23 and 57, through their respective dependence from claims 22 and 56, require both maintenance data and static data. It is not clear how these individual elements are delineated in the *Anshel et al.* invention. That is, a close reading of the disclosure of the invention, and of the cited passage in particular, does not reveal a correspondence of disclosed elements with claimed elements. The lack of detail in the Examiner's assertions does not assist in making this identification. In any case, *Anshel et al.* does not disclose both maintenance and static data, which is updated, as components of a key split that is randomized with other key splits to produce a key, as recited in the claims. As previously discussed, *Hirsch* also fails to disclose these elements of the claimed invention. Thus, no combination of the teachings of the cited references could render obvious the invention recited in claims 23 and 57. The rejection should be withdrawn.

Claim 24 recites that the means for updating the static data includes means for modifying a prime number divisor of the static data. Claim 58 recites that updating the static data includes modifying a prime number divisor of the static data.

Regarding claims 24 and 58, the Examiner stated that *Anshel et al.* illustrate that updating the static data includes modifying a prime number divisor of the static data, citing column 11, lines 8-25 and Figure 8, item 71. In the passage cited by the Examiner, *Anshel et al.* describe use of randomly-selected prime numbers to generate a list of Jacobi symbols and, according to a public key, choosing a subset of the symbols to encrypt a single input bit. The Jacobi symbols and the public key both feature terms having prime number divisors. See column 11, lines 8-53. The described process is not connected in any way to anything that the Examiner has identified as being static data, or to the

updating of this static data. In contrast, claims 24 and 58 recite updating the static data by modifying a prime number divisor of the static data (a basis for a random key split).

Demonstrating that it is known to use prime number divisors in an application related to cryptography is not enough to suggest to one of ordinary skill in the art that modification of a prime number divisor of static data basis of a key split is beneficial in generating a key by randomizing a number of key splits including the random key split, particularly in view of other references that do not even disclose the use of key splits. It is clear that no combination of the teachings of the cited references is suggested in any of the references, or is even possible given their respective disclosed features. Even if the disclosed features were somehow combined, the resulting amalgamation would not include all the features of the claimed invention, and therefore could not possibly render obvious the invention recited in claims 24 and 58. The rejection of these claims, therefore, should be withdrawn.

NINTH ARGUMENT

The Examiner's rejection of claims 19 and 53 as being unpatentable over *Hirsch*, in view of *Anshel et al.* and *Albert et al.*, is not supported by the teachings of the references. First, it would be improper to attempt to combine the teachings of the cited references, which do not disclose or suggest any motivation for such combination, or even how any such modification could be accomplished. Further, even if such combination were possible and proper, any resulting system still would not include all the elements of the claimed invention.

A. THE EXAMINER'S POSITION

The Examiner asserted that *Hirsch* in view of *Anshel et al.* disclose the claimed cryptographic key split combiner and process. The Examiner further asserted that *Anshel et al.* describe a means for generating a pseudorandom sequence based on maintenance data (citing column 8, lines 8-15). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* in view of *Anshel et al.* with a means for generating a pseudorandom sequence of *Anshel et al.* for “very simple and highly secure authentication” (citing column 8, lines 8 and 9). The Examiner acknowledged that these references do not explicitly teach a random sequence. The Examiner asserted that *Albert et al.* specify a random sequence (citing column 1, lines 51-67, and column 2, lines 1 and 2). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* in view of *Anshel et al.* with a means for generating a random sequence of *Albert et al.* to “increase the quality of random numbers with respect to their predictability and their functional link” (citing column 1, lines 66 and 67, and column 2, lines 1 and 2). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

B. APPELLANT'S POSITION

1. Overview of Hirsch

Hirsch discloses a software data protection mechanism, which includes an apparatus for generating a key. The description of the key generating apparatus is set forth in detail from column 3, line 47 through column 5, line 28, with reference to Figs. 1A-C. First, a 32-bit binary value is loaded into an input register 12. Each bit of the 32-bit value is provided as an input to a corresponding scrambler array container 140-n, where n ranges from 0 to 31. Each container 140-n in the scrambler array 14 determines whether the input bit of that container is passed to a respective bit position of a 32-bit output register 18, or whether the complement of that bit is passed instead.

The control for determining whether the input bit or its complement is passed is determined by every fifth bit of a pseudorandom sequence that is generated by a pseudorandom number generator 16. This generator takes a single seed and generates a single sequence that is loaded serially into the containers of the scrambler array 14, such that five bits are stored in each container. Because there are 32 containers, a 5-bit number also designates each container (bit position) (00000, 00001, 00010, 00011, ... , 11101, 11110, 11111). The result of an XOR operation on the LSB of the bit position and the LSB of the 5-bit pseudorandom sequence segment stored in the container determines whether the input bit or its complement will be provided to the output register. Because the LSB of the bit position alternates in the sequence of scrambler array containers, the XOR result for all even-numbered containers will be the pseudorandom number LSB, and the XOR result for all odd-numbered containers will be the pseudorandom number LSB

complement. Thus, the XOR result, and therefore the determination of whether the input bit or its complement is provided to the output register, is determined solely by the pseudorandom sequence. The actual bit position of the output register to which the respective "scrambled" input value bits is provided is determined by the actual value of the pseudorandom number, although the particular correspondence is not disclosed by *Hirsch*. Thus, the original 32-bit value is modified to provide another 32-bit value.

The 32-bit value now stored in the output register 18 is encoded by mapping the values of grouped bits according to a table. That is, the 32 bits of output data are separated into four groups of eight bits each. Each 8-bit group is stored in a set of two overlapping 5-bit registers 200-1a (bits 0-4) and 200-1b (bits 4-8). The outputs of these registers (eight 5-bit values) are provided to a fixed alphanumeric table 204, which maps the input values to provide eight alphanumeric output values. The sequence consisting of these output values is the key, which is stored in a user key register 24.

2. Overview of *Anshel et al.*

Anshel et al. disclose a cryptographically secure sequence generator used for pseudorandom sequence generation, authentication, and public key transfer and encryption. The public keys are pseudorandom sequences based on zeta one-way functions, namely quadratic signatures that are special sequences of Jacobi symbols.

3. Overview of *Albert et al.*

Albert et al. merely disclose a random number generator, particularly for use in a cryptographic system.

4. Distinction between the subject matter of claims 19 and 53 and the *Hirsch*, *Anshel et al.*, and *Albert et al.* disclosures

Claim 19 depends from claim 1, through intervening claim 18. Thus, claim 19 recites a cryptographic key split combiner, which includes a plurality of key split generators for generating cryptographic key splits, and a key split randomizer for randomizing the cryptographic key splits to produce a cryptographic key. The plurality of key split generators includes a console split generator for generating a console key split based on maintenance data. The console split generator includes means for generating a random sequence based on the maintenance data.

Claim 53 depends from claim 35, through intervening claim 52. Thus, claim 53 recites a process for forming cryptographic keys, which includes generating a plurality of cryptographic key splits from seed data, and randomizing the cryptographic key splits to produce a cryptographic key. Generating a plurality of cryptographic key splits includes generating a console key split based on maintenance data, which includes generating a random sequence based on the maintenance data.

Claims 19 and 53 require generating a console key split, among other key splits that are to be randomized to produce a key. The console key split is based on maintenance data, and the console split generator (or generating the console split) includes generation of a random sequence based on the maintenance data. As previously discussed, *Hirsch* does not disclose any of these elements. *Anshel et al.* describe generating a public key, from a pseudorandom sequence that is based on a state value. In the passage cited by the Examiner, *Anshel et al.* describe use of randomly-selected prime numbers to generate a list of Jacobi symbols and, according to a public key, chooses a

subset of the symbols to encrypt a single input bit. Thus, the passage describes the encryption algorithm to which the key is applied; it does not describe a key split. *Anshel et al.* do not disclose a console key split based on maintenance data, which is randomized with other key splits to produce a key.

Albert et al. merely describe a circuit for generating a random sequence. There is no suggestion by *Albert et al.* that the disclosed random number generator should or could be used as part of a console split generator in a cryptographic key split combiner as recited in claims 19 and 53. Further, *Albert et al.* does not overcome the deficiencies of *Hirsch* in disclosing the combiner recited in claim 1. That is, *Hirsch* does not disclose a combiner that takes separate key splits, generated based on separate seed values, and randomizes the splits to provide a key, as recited in claim 1. *Albert et al.*, by merely describing a random number generator, do not provide *Hirsch* with a means for generating separate key splits from separate seed values, or for randomizing separate splits to provide a key.

The Examiner does not describe in any detail how the teachings of *Albert et al.* could be combined with the *Hirsch* system, but claims 19 and 53 require that the console split generator generate the pseudorandom sequence based on the maintenance data. Somehow substituting the *Albert et al.* generator for the pseudorandom sequence generator of *Anshel et al.* does not overcome any of the previously-mentioned deficiencies, and is not suggested in any of the references.

Thus, none of the references discloses or suggests all the elements of claims 19 and 53, that is, generating a console key split, among other key splits, that are to be randomized to produce a key, where the console key split is based on maintenance data,

and the console split generator (or generating the console split) includes generation of a random sequence based on the maintenance data. Therefore, no combination of the teachings of the cited references could render obvious the invention recited in claims 19 and 53. The rejection should be withdrawn.

TENTH ARGUMENT

The Examiner's rejection of claims 25, 27-31, 59, and 61-65 as being unpatentable over *Hirsch*, in view of *Tomko et al.*, is not supported by the teachings of the references. First, it would be improper to attempt to combine the teachings of the cited references, which do not disclose or suggest any motivation for such combination, or even how any such modification could be accomplished. Further, even if such combination were possible and proper, any resulting system still would not include all the elements of the claimed invention.

A. THE EXAMINER'S POSITION

Regarding claims 25 and 59, the Examiner asserted that *Hirsch* teaches the claimed cryptographic key split combiner and process. The Examiner acknowledged that *Hirsch* does not suggest generating a biometric key split based on biometric data. The Examiner asserted that *Tomko et al.* "elaborate on" a biometric split generator for generating a biometric key split based on biometric data (citing column 2, lines 2-20). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with a biometric split generator for generating a biometric key split based on biometric

data of *Tomko et al.* to “have [a] secure, yet readily available private key” (citing column 1, lines 58-60). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 27 and 61, the Examiner asserted that *Tomko et al.* further disclose a means for generating a pseudorandom sequence based on the biometric data (citing column 2, lines 2-12). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with a means for generating a pseudorandom sequence based on the biometric data of *Tomko et al.* to “have [a] secure, yet readily available private key” (citing column 1, lines 58-60). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 28 and 62, the Examiner asserted that *Tomko et al.* “delineate” a means for generating a key split based on biometric data vectors and on biometric combiner data (citing column 3, lines 56-67). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with a means for generating a key split based on biometric data vectors and on biometric combiner data of *Tomko et al.* to “have [a] secure, yet readily available private key” (citing column 1, lines 58-60). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 29 and 63, the Examiner asserted that *Tomko et al.* explain a means for generating a key split based on biometric data and on static data (citing column

3, lines 56-67). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with a means for generating a key split based on biometric data and on static data of *Tomko et al.* to “have [a] secure, yet readily available private key” (citing column 1, lines 58-60). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 30 and 64, the Examiner asserted that *Tomko et al.* illustrate updating the static data (citing column 7, lines 33-39, and Fig. 1, elements 43 and 44). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with updating the static data of *Tomko et al.* for “enrolling an individual” (citing column 7, lines 33-36). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

Regarding claims 31 and 65, the Examiner asserted that *Tomko et al.* “elaborate” that the means for updating the static data includes means for modifying a prime number divisor of the static data (citing column 7, lines 45-67, and column 8, lines 1-12). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* with a means for modifying a prime number divisor of the static data of *Tomko et al.* so that “the subscriber can later reproduce the static data” (citing column 8, lines 22-24). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

B. APPELLANT'S POSITION

1. Overview of *Hirsch*

Hirsch discloses a software data protection mechanism, which includes an apparatus for generating a key. The description of the key generating apparatus is set forth in detail from column 3, line 47 through column 5, line 28, with reference to Figs. 1A-C. First, a 32-bit binary value is loaded into an input register 12. Each bit of the 32-bit value is provided as an input to a corresponding scrambler array container 140-n, where n ranges from 0 to 31. Each container 140-n in the scrambler array 14 determines whether the input bit of that container is passed to a respective bit position of a 32-bit output register 18, or whether the complement of that bit is passed instead.

The control for determining whether the input bit or its complement is passed is determined by every fifth bit of a pseudorandom sequence that is generated by a pseudorandom number generator 16. This generator takes a single seed and generates a single sequence that is loaded serially into the containers of the scrambler array 14, such that five bits are stored in each container. Because there are 32 containers, a 5-bit number also designates each container (bit position) (00000, 00001, 00010, 00011, ..., 11101, 11110, 11111). The result of an XOR operation on the LSB of the bit position and the LSB of the 5-bit pseudorandom sequence segment stored in the container determines whether the input bit or its complement will be provided to the output register. Because the LSB of the bit position alternates in the sequence of scrambler array containers, the XOR result for all even-numbered containers will be the pseudorandom number LSB, and the XOR result for all odd-numbered containers will be the pseudorandom number LSB

complement. Thus, the XOR result, and therefore the determination of whether the input bit or its complement is provided to the output register, is determined solely by the pseudorandom sequence. The actual bit position of the output register to which the respective "scrambled" input value bits is provided is determined by the actual value of the pseudorandom number, although the particular correspondence is not disclosed by *Hirsch*. Thus, the original 32-bit value is modified to provide another 32-bit value.

The 32-bit value now stored in the output register 18 is encoded by mapping the values of grouped bits according to a table. That is, the 32 bits of output data are separated into four groups of eight bits each. Each 8-bit group is stored in a set of two overlapping 5-bit registers 200-1a (bits 0-4) and 200-1b (bits 4-8). The outputs of these registers (eight 5-bit values) are provided to a fixed alphanumeric table 204, which maps the input values to provide eight alphanumeric output values. The sequence consisting of these output values is the key, which is stored in a user key register 24.

2. Overview of *Tomko et al.*

Tomko et al. disclose a fingerprint-controlled public key cryptographic system. *Tomko et al.* utilize data derived from a user's fingerprint to generate a private key to be used in encrypting and decrypting messages. The fingerprint data is provided as an input to a pseudorandom sequence generator to generate the key. No other components are included in the key.

3. Distinction between the subject matter of claims 25, 27-31, 59, and 61-65 and the Hirsch and Tomko et al. disclosures

Claims 25 and 27-31 depend from claim 1, and therefore require a cryptographic key split combiner, which includes a plurality of key split generators for generating cryptographic key splits, and a key split randomizer for randomizing the cryptographic key splits to produce a cryptographic key. Claims 59 and 61-65 depend from claim 35, and therefore require a process for forming cryptographic keys, which includes generating a plurality of cryptographic key splits from seed data, and randomizing the cryptographic key splits to produce a cryptographic key.

Claim 25 recites that the plurality of key split generators includes a biometric split generator for generating a biometric key split based on biometric data. Claim 59 recites that generating the plurality of key splits includes generating a biometric key split based on biometric data.

Regarding claims 25 and 59, the Examiner stated that *Tomko et al.* elaborate on a biometric split generator for generating a biometric key split based on biometric data. *Tomko et al.* disclose a fingerprint-controlled public key cryptographic system. *Tomko et al.* utilize data derived from a user's fingerprint to generate a private key to be used in encrypting and decrypting messages. The fingerprint data is provided as an input to a pseudorandom sequence generator to generate the key. No other components are included in the key, and therefore the biometric data used in the *Tomko et al.* system is not used to generate a biometric key split, as recited in the claims, but rather is used to generate the key itself, that is, as a seed for generating the sequence. No other component is randomized with the biometric data to derive the key. As discussed above, *Hirsch* does not disclose a randomizer for combining key splits derived from individual seeds to

generate a key, as recited in claims 1 and 35. Therefore, there is no suggestion or possibility of using the biometric-derived key of *Tomko et al.* as a split in the *Hirsch* system. Thus, no combination of the teachings of the cited references could render obvious the invention recited in claims 25 and 59. The rejection of these claims should be withdrawn.

Claim 27 recites that the biometric split generator includes means for generating a pseudorandom sequence based on the biometric data. Claim 61 recites that generating the biometric split includes generating a pseudorandom sequence based on the biometric data.

Regarding claims 27 and 61, the Examiner stated that *Tomko et al.* disclose means for generating a pseudorandom sequence based on the biometric data. However, claims 27 and 61 require biometric key split generation, which, as noted above in discussing the rejection of claims 25 and 59, is not disclosed by *Tomko et al.* As discussed above, *Hirsch* does not disclose a randomizer for combining key splits derived from individual seeds to generate a key, as recited in claims 1 and 35, from which claims 27 and 61 depend. Therefore, *Hirsch* cannot accept the *Tomko et al.* biometric key as a split for randomizing with its own key to form a further cryptographic key. Thus, no combination of the teachings of the cited references could render obvious the invention recited in claims 27 and 61. The rejection of these claims should be withdrawn.

Claim 28 recites that the biometric split generator includes means for generating a key split based on biometric data vectors and on biometric combiner data. Claim 62 recites that generating the biometric split includes generating a key split based on biometric data vectors and on biometric combiner data.

Regarding claims 28 and 62, the Examiner stated that *Tomko et al.* delineate means for generating a key split based on biometric data vectors and on biometric combiner data. The Examiner did not specifically identify the data vectors or the combiner data. Regardless, *Tomko et al.* do not disclose generating a key split at all. Rather, *Tomko et al.* disclose generating a key based solely on biometric data as a seed value, not on randomized splits. As discussed above, *Hirsch* does not disclose a randomizer for combining key splits derived from individual seeds to generate a key, as recited in claims 1 and 35, from which claims 28 and 62 depend, and cannot process any *Tomko et al.* output as a key split.. Thus, no combination of the teachings of the cited references could render obvious the invention recited in claims 28 and 62. The rejection of these claims should be withdrawn.

Claim 29 recites that the biometric split generator includes means for generating a key split based on the biometric data and on static data. Claim 63 recites that generating the biometric split includes generating a key split based on the biometric data and on static data.

Regarding claims 29 and 63, the Examiner stated that *Tomko et al.* explain a means for generating a key split based on biometric data and on static data. The Examiner cited the same passage of *Tomko et al.* as that cited in rejecting claims 28 and 62, but did not specifically identify what he considers to be the static data. That is, the Examiner cited column 3, lines 56-67 as disclosing static data in addition to biometric data. However, this passage only discloses calculation of a digital Fourier transform from the biometric data (fingerprint information) itself. Thus, this is merely disclosure of more (derivative) biometric data.

As discussed above, *Hirsch* does not disclose a randomizer for combining key splits derived from individual seeds to generate a key, as recited in claims 1 and 35, from which claims 29 and 63 depend. Further, *Tomko et al.* do not disclose generating a key split at all. Rather, *Tomko et al.* disclose generating a key based solely on biometric data as a seed value, not on randomized splits. Thus, no combination of the teachings of the cited references could render obvious the invention recited in claims 29 and 63. The rejection of these claims should be withdrawn.

Claim 30 recites that the combiner of claim 29 further includes means for updating the static data. Claim 64 recites that the process of claim 63 further includes updating the static data.

Regarding claims 30 and 64, the Examiner stated that *Tomko et al.* illustrate updating the static data. However, *Tomko et al.* do not disclose generating a biometric key split, or any key split at all, as recited in the claims. Rather, *Tomko et al.* disclose generating a key based solely on biometric data as a seed value, not on randomized splits. The Examiner cited column 7, lines 33-39, as disclosing an update of the static data. However, this passage merely discloses the initial assignment of a unique user number based on a Fourier transform of fingerprint data. Thus, this is not static data (it is modified biometric data), and it is not an updated value (it is the initial determination of the unique number). As discussed above, *Hirsch* does not disclose a randomizer for combining key splits derived from individual seeds to generate a key, as recited in claims 1 and 35, from which claims 30 and 64 depend, and does not even suggest randomizing a received key with the generated key as splits to form another key. Thus, no combination

of the teachings of the cited references could render obvious the invention recited in claims 30 and 64. The rejection of these claims should be withdrawn.

Claim 31 recites that the means for updating the static data includes means for modifying a prime number divisor of the static data. Claim 65 recites that updating the static data includes modifying a prime number divisor of the static data.

Regarding claims 31 and 65, the Examiner stated that *Tomko et al.* elaborate that the means for updating the static data includes means for modifying the prime number divisor of the static data, citing column 7, line 45 - column 8, line 12. However, this passage describes the procedure used to derive an array b that is related to the unique number u , which is derived from the Fourier transform of the user's fingerprint data. That is, the passage describes the modular mathematics used to derive the coefficients b that determine u . Prime numbers are not mentioned at all, and certainly not in terms of modifying the prime number divisor of any static data. No primes are used, and the only data used is derived from the biometric data.

Further, *Tomko et al.* do not disclose generating a key split at all. Rather, *Tomko et al.* disclose generating a key based solely on biometric data as a seed value, not on randomized splits. As discussed above, *Hirsch* does not disclose a randomizer for combining key splits derived from individual seeds to generate a key, as recited in claims 1 and 35, from which claims 31 and 65 depend. Thus, no combination of the teachings of the cited references could render obvious the invention recited in claims 31 and 65. The rejection of these claims should be withdrawn.

ELEVENTH ARGUMENT

The Examiner's rejection of claims 26 and 60 as being unpatentable over *Hirsch*, in view of *Tomko et al.* and *Albert et al.*, is not supported by the teachings of the references. First, it would be improper to attempt to combine the teachings of the cited references, which do not disclose or suggest any motivation for such combination, or even how any such modification could be accomplished. Further, even if such combination were possible and proper, any resulting system still would not include all the elements of the claimed invention.

A. THE EXAMINER'S POSITION

The Examiner asserted that *Hirsch* in view of *Tomko et al.* disclose the cryptographic key split combiner and process of combining of claim 25. The Examiner further asserted that *Tomko et al.* discuss a means for generating a pseudorandom sequence based on the biometric data (citing column 2, lines 2-12). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* in view of *Tomko et al.* with a means for generating a pseudorandom sequence based on the biometric data of *Tomko et al.* to "have [a] secure, yet readily available private key" (citing column 1, lines 58-60). The Examiner acknowledged that these references do not explicitly teach a random sequence. The Examiner asserted that *Albert et al.* specify a random sequence (citing column 1, lines 51-67, and column 2, lines 1 and 2). The Examiner concluded that it would have been obvious to one of ordinary skill in the art to combine the cryptographic key split combiner and process of combining of *Hirsch* in

view of *Tomko et al.* with a means for generating a random sequence of *Albert et al.* to “increase the quality of random numbers with respect to their predictability and their functional link” (citing column 1, lines 66 and 67, and column 2, lines 1 and 2). The Examiner provided no details as to the motivation behind any such combination, or how such combination would be accomplished.

B. APPELLANT’S POSITION

1. Overview of Hirsch

Hirsch discloses a software data protection mechanism, which includes an apparatus for generating a key. The description of the key generating apparatus is set forth in detail from column 3, line 47 through column 5, line 28, with reference to Figs. 1A-C. First, a 32-bit binary value is loaded into an input register 12. Each bit of the 32-bit value is provided as an input to a corresponding scrambler array container 140-n, where n ranges from 0 to 31. Each container 140-n in the scrambler array 14 determines whether the input bit of that container is passed to a respective bit position of a 32-bit output register 18, or whether the complement of that bit is passed instead.

The control for determining whether the input bit or its complement is passed is determined by every fifth bit of a pseudorandom sequence that is generated by a pseudorandom number generator 16. This generator takes a single seed and generates a single sequence that is loaded serially into the containers of the scrambler array 14, such that five bits are stored in each container. Because there are 32 containers, a 5-bit number also designates each container (bit position) (00000, 00001, 00010, 00011, ... , 11101,

11110, 11111). The result of an XOR operation on the LSB of the bit position and the LSB of the 5-bit pseudorandom sequence segment stored in the container determines whether the input bit or its complement will be provided to the output register. Because the LSB of the bit position alternates in the sequence of scrambler array containers, the XOR result for all even-numbered containers will be the pseudorandom number LSB, and the XOR result for all odd-numbered containers will be the pseudorandom number LSB complement. Thus, the XOR result, and therefore the determination of whether the input bit or its complement is provided to the output register, is determined solely by the pseudorandom sequence. The actual bit position of the output register to which the respective “scrambled” input value bits is provided is determined by the actual value of the pseudorandom number, although the particular correspondence is not disclosed by *Hirsch*. Thus, the original 32-bit value is modified to provide another 32-bit value.

The 32-bit value now stored in the output register 18 is encoded by mapping the values of grouped bits according to a table. That is, the 32 bits of output data are separated into four groups of eight bits each. Each 8-bit group is stored in a set of two overlapping 5-bit registers 200-1a (bits 0-4) and 200-1b (bits 4-8). The outputs of these registers (eight 5-bit values) are provided to a fixed alphanumeric table 204, which maps the input values to provide eight alphanumeric output values. The sequence consisting of these output values is the key, which is stored in a user key register 24.

2. Overview of *Tomko et al.*

Tomko et al. disclose a fingerprint-controlled public key cryptographic system. *Tomko et al.* utilize data derived from a user’s fingerprint to generate a private key to be

used in encrypting and decrypting messages. The fingerprint data is provided as an input to a pseudorandom sequence generator to generate the key. No other components are included in the key.

3. Overview of *Albert et al.*

Albert et al. merely disclose a random number generator, particularly for use in a cryptographic system.

4. Distinction between the subject matter of claims 26 and 60 and the *Hirsch*, *Tomko et al.*, and *Albert et al.* disclosures

Claim 26 depends from claim 1, through intervening claim 25. Thus, claim 26 recites a cryptographic key split combiner, which includes a plurality of key split generators for generating cryptographic key splits, and a key split randomizer for randomizing the cryptographic key splits to produce a cryptographic key. The plurality of key split generators includes a biometric split generator for generating a biometric key split based on biometric data. The biometric split generator includes means for generating a random sequence based on the biometric data.

Claim 60 depends from claim 35, through intervening claim 59. Thus, claim 60 recites a process for forming cryptographic keys, which includes generating a plurality of cryptographic key splits from seed data, and randomizing the cryptographic key splits to produce a cryptographic key. Generating a plurality of cryptographic key splits includes generating a biometric key split based on biometric data, which includes generating a random sequence based on the biometric data.

As thoroughly noted above in discussing the other rejections, *Hirsch* does not disclose the combiner recited in claim 1, or the process of claim 35. *Tomko et al.* do not disclose generating a key split. Rather, *Tomko et al.* disclose generating an actual key based solely on biometric data (including data derived solely from the biometric data), not on randomized splits. *Albert et al.* merely describe a random sequence generator. There is no suggestion by *Albert et al.* that the disclosed random number generator should or could be used as part of a random split generator in a cryptographic key split combiner as recited in claims 1 and 35. *Albert et al.*, by merely describing a random number generator, do not provide *Hirsch* with a means for generating separate key splits from separate seed values, or for randomizing separate splits to provide a key.

The Examiner implied that the *Albert et al.* random number generator could be substituted for the *Tomko et al.* pseudorandom number generator to provide less predictability. However, this would not overcome any of the previously-mentioned deficiencies in the teachings of the cited references. No combination of the teachings in this manner is suggested, and any such combination would not disclose or suggest generating a plurality of key splits from seed data, which splits are randomized to form a key.

As discussed above, *Hirsch* does not disclose a randomizer for combining key splits derived from individual seeds to generate a key, as recited in claims 1 and 35, from which claims 26 and 60 depend. Thus, no combination of the teachings of the cited references could render obvious the invention recited in claims 26 and 60. The rejection of these claims should be withdrawn.

SUMMARY AND FINAL CONCLUSION

The application includes two independent claims. Claim 1 recites a cryptographic key split combiner, which includes a plurality of key split generators for generating cryptographic key splits, and a key split randomizer for randomizing the cryptographic key splits to produce a cryptographic key. Each of said key split generators includes means for generating key splits from seed data. Claim 35 recites a process for forming cryptographic keys, which includes generating a plurality of cryptographic key splits from seed data, and randomizing the cryptographic key splits to produce a cryptographic key.

The Examiner's primary reference, *Hirsch*, discloses a system for producing a key. A single input value is modified according to a pseudorandom sequence. The modified value is mapped according to a fixed table in order to generate the key. The Examiner's primary reference fails to disclose the claimed invention. None of the Examiner's secondary references make up for the deficiencies of the *Hirsch* disclosure. *Albert et al.* merely disclose a random number generator. *Thomlinson et al.* disclose a pseudorandom sequence generator bases its seed in part on chronological data. *Ming et al.* disclose a cable television decoder that descrambles a television signal according to a pseudorandom sequence if an encrypted, embedded access code matches a fixed sequence at the receiver. *Anshel et al.* disclose a pseudorandom sequence generator used for public key transfers. *Tomko et al.* disclose a cryptographic system that generates keys based solely on biometric data.

All of the references cited by the Examiner relate to cryptography. However, none of the references suggests combination of their respective features such that the

Hirsch system would process a plurality of key splits, generated by a plurality of key split generators, by randomizing the splits to form the key. Random functions are described in all of the references, but not in the manner recited in the claims. Encryption of data is disclosed in some of the references, but not in the manner recited in the claims. Each of the cited systems operates in a manner that is particular to that disclosed system. Combination of the teachings of the references is not practical and, consequently, not suggested in the references. Taking bits and pieces of the disclosed systems, out of context and with no suggested motivation, and trying to assemble these pieces to form any working cryptographic system, is not feasible. Such assembly certainly cannot be accomplished in a manner that would result in the claimed invention. A person of ordinary skill in the art is provided with no guidance as to how or why to attempt such disassembly and reassembly. The Examiner's rejections themselves only provide vague hints as to how he believes that the various combinations would be accomplished.

An analysis of the claimed invention and the cited references leads to certain indisputable conclusions. The primary reference, *Hirsch*, does not anticipate the claimed invention. Combination of the teachings of the references in an attempt to render obvious the claimed invention is not suggested in the references, is taught against in some cases, and is therefore improper. Even if such combination were proper, no combination of the teachings of the references could result in the claimed invention.

Based on the foregoing, it is submitted that all rejections have been successfully traversed. It is therefore requested that the claims be allowed and the case passed to issue.

A petition for extension of time is filed herewith. A check in payment of the fee for the extension is also enclosed. If the check is missing or made out for an insufficient amount, please charge our deposit account, No. 18-0002, and notify us accordingly.

Respectfully submitted,

February 12, 2001

Date



Thomas M. Champagne
Registration No. 36,478
RABIN & CHAMPAGNE, P.C.
Customer No. 23995
(202) 659-1915
(202) 659-1898 fax

TMC:lep

Certification under 37 CFR 1.8

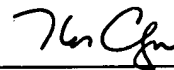
February 12, 2001

Date of Deposit

I hereby certify that this Appeal Brief, along with a petition for extension of time, and check in payment of the petition fee, is being deposited with the United States Postal Service as First Class Mail under 37 CFR §1.8 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Thomas M. Champagne

Typed or printed name of person
mailing Appeal Brief



Signature of person mailing
Appeal Brief

APPENDIX A

What is claimed is:

1. A cryptographic key split combiner, comprising:
 - a) a plurality of key split generators for generating cryptographic key splits; and
 - b) a key split randomizer for randomizing the cryptographic key splits to produce a cryptographic key;
 - c) wherein each of said key split generators includes means for generating key splits from seed data.
2. The cryptographic key split combiner of claim 1, wherein said plurality of key split generators includes a random split generator for generating a random key split based on reference data.
3. The cryptographic key split combiner of claim 2, wherein said random split generator includes means for generating a random sequence based on the reference data.
4. The cryptographic key split combiner of claim 2, wherein said random split generator includes means for generating a pseudorandom sequence based on the reference data.
5. The cryptographic key split combiner of claim 2, wherein said random split generator includes means for generating a key split based on the reference data and on chronological data.

6. The cryptographic key split combiner of claim 2, wherein said random split generator includes means for generating a key split based on the reference data and on static data.

7. The cryptographic key split combiner of claim 6, further including means for updating the static data.

8. The cryptographic key split combiner of claim 7, wherein the means for updating the static data includes means for modifying a prime number divisor of the static data.

9. The cryptographic key split combiner of claim 1, wherein said plurality of key split generators includes a token split generator for generating a token key split based on label data.

10. The cryptographic key split combiner of claim 9, further comprising means for reading the label data from a storage medium.

11. The cryptographic key split combiner of claim 9, wherein the label data includes user authorization data.

12. The cryptographic key split combiner of claim 9, wherein said token split generator includes means for generating a random sequence based on the label data.

13. The cryptographic key split combiner of claim 9, wherein said token split generator includes means for generating a pseudorandom sequence based on the label data.

14. The cryptographic key split combiner of claim 9, wherein said token split generator includes means for generating a key split based on the label data and on organization data.

15. The cryptographic key split combiner of claim 9, wherein said token split generator includes means for generating a key split based on the label data and on static data.

16. The cryptographic key split combiner of claim 15, further including means for updating the static data.

17. The cryptographic key split combiner of claim 16, wherein the means for updating the static data includes means for modifying a prime number divisor of the static data.

18. The cryptographic key split combiner of claim 1, wherein said plurality of key split generators includes a console split generator for generating a console key split based on maintenance data.

19. The cryptographic key split combiner of claim 18, wherein said console split generator includes means for generating a random sequence based on the maintenance data.

20. The cryptographic key split combiner of claim 18, wherein said console split generator includes means for generating a pseudorandom sequence based on the maintenance data.

21. The cryptographic key split combiner of claim 18, wherein said console split generator includes means for generating a key split based on previous maintenance data and on current maintenance data.

22. The cryptographic key split combiner of claim 18, wherein said console split generator includes means for generating a key split based on the maintenance data and on static data.

23. The cryptographic key split combiner of claim 22, further including means for updating the static data.

24. The cryptographic key split combiner of claim 22, wherein the means for updating the static data includes means for modifying a prime number divisor of the static data.

25. The cryptographic key split combiner of claim 1, wherein said plurality of key split generators includes a biometric split generator for generating a biometric key split based on biometric data.

26. The cryptographic key split combiner of claim 25, wherein said biometric split generator includes means for generating a random sequence based on the biometric data.

27. The cryptographic key split combiner of claim 25, wherein said biometric split generator includes means for generating a pseudorandom sequence based on the biometric data.

28. The cryptographic key split combiner of claim 25, wherein said biometric split generator includes means for generating a key split based on biometric data vectors and on biometric combiner data.

29. The cryptographic key split combiner of claim 25, wherein said biometric split generator includes means for generating a key split based on the biometric data and on static data.

30. The cryptographic key split combiner of claim 29, further including means for updating the static data.

31. The cryptographic key split combiner of claim 30, wherein the means for updating the static data includes means for modifying a prime number divisor of the static data.

32. The cryptographic key split combiner of claim 1, wherein the cryptographic key is a stream of symbols.

33. The cryptographic key split combiner of claim 1, wherein the cryptographic key is at least one symbol block.

34. The cryptographic key split combiner of claim 1, wherein the cryptographic key is a key matrix.

35. A process for forming cryptographic keys, comprising:

- a) generating a plurality of cryptographic key splits from seed data; and
- b) randomizing the cryptographic key splits to produce a cryptographic key.

36. The process of claim 35, wherein generating a plurality of cryptographic key splits includes generating a random key split based on reference data.

37. The process of claim 36, wherein generating a random key split includes generating a random sequence based on the reference data.

38. The process of claim 36, wherein generating a random key split includes generating a pseudorandom sequence based on the reference data.

39. The process of claim 36, wherein generating a random key split includes generating a key split based on the reference data and on chronological data.

40. The process of claim 36, wherein generating a random key split includes generating a key split based on the reference data and on static data.

41. The process of claim 40, further including updating the static data.

42. The process of claim 41, wherein updating the static data includes modifying a prime number divisor of the static data.

43. The process of claim 35, wherein generating a plurality of cryptographic key splits includes generating a token key split based on label data.

44. The process of claim 43, further comprising reading the label data from a storage medium.

45. The process of claim 43, wherein the label data includes user authorization data.

46. The process of claim 43, wherein generating a token key split includes generating a random sequence based on the label data.
47. The process of claim 43, wherein generating a token key split includes generating a pseudorandom sequence based on the label data.
48. The process of claim 43, wherein generating a token key split includes generating a key split based on the label data and on organization data.
49. The process of claim 43, wherein generating a token key split includes generating a key split based on the label data and on static data.
50. The process of claim 49, further including updating the static data.
51. The process of claim 50, wherein updating the static data includes modifying a prime number divisor of the static data.
52. The process of claim 35, wherein generating a plurality of cryptographic key splits includes generating a console key split based on maintenance data.
53. The process of claim 52, wherein generating a console key split includes generating a random sequence based on the maintenance data.

54. The process of claim 52, wherein generating a console key split includes generating a pseudorandom sequence based on the maintenance data.

55. The process of claim 52, wherein generating a console key split includes generating a key split based on previous maintenance data and on current maintenance data.

56. The process of claim 52, wherein generating a console key split includes generating a key split based on the maintenance data and on static data.

57. The process of claim 56, further including updating the static data.

58. The process of claim 56, wherein the updating the static data includes modifying a prime number divisor of the static data.

59. The process of claim 35, wherein generating a plurality of cryptographic key splits includes generating a biometric key split based on biometric data.

60. The process of claim 59, wherein generating a biometric key split includes generating a random sequence based on the biometric data.

61. The process of claim 59, wherein generating a biometric key split includes generating a pseudorandom sequence based on the biometric data.

62. The process of claim 59, wherein generating a biometric key split includes generating a key split based on biometric data vectors and on biometric combiner data.

63. The process of claim 59, wherein generating a biometric key split includes generating a key split based on the biometric data and on static data.

64. The process of claim 63, further including updating the static data.

65. The process of claim 63, wherein updating the static data includes modifying a prime number divisor of the static data.

66. A cryptographic key, formed by the process of claim 35.

67. The cryptographic key of claim 66, including a stream of symbols.

68. The cryptographic key of claim 66, including at least one symbol block.

69. The cryptographic key of claim 66, including a key matrix.